Manuel 3

Circuits d'Interface Microcontrôleur

www.picaxe.com

Table des Matières

Traduction : Hervé BLOREC – v.1.03-2016

À propos de ce manuel

S'il vous plaît notez qu'une version plus récente de ce manuel est en préparation, consultez le site : www.picaxe.co.uk pour obtenir la dernière version.

Le manuel PICAXE est divisé en trois sections séparées :

Section 1 – Débuter	(picaxe_manual1_fr.pdf)
Section 2 – Commandes du BASIC	(picaxe_manual2_fr.pdf)
Section 3 – Circuits d'Interfaçage Microcontrôleur	(picaxe_manual3_fr.pdf)

Cette troisième section fournit des circuits d'interfaçage généraux pour les microcon-trôleur et des exemples de programmes, pour la plupart des transducteurs communs d'entrée/sortie utilisés au sein des circuits de microcontrôleur.

Pour une information générale pour débuter avec le système PICAXE voyez le manuel 1. Il n'y a pas de connaissances prioritaires à connaître sur les microcontrôleurs. Une série de tutoriaux introduit les fonctions principales du système.

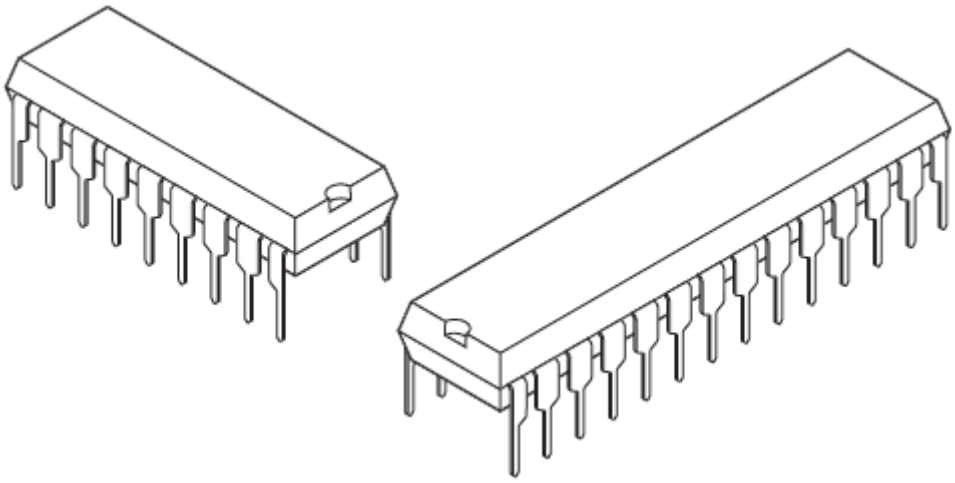
Pour plus d'informations spécifiques, de syntaxe et d'exemples de Commande BASIC voir le manuel 2 « Commandes BASIC ».

Le logiciel utilisé pour programmer les PICAXE s'appelle PE6 (Programming Editor 6). Ce logiciel est en téléchargement libre à l'adresse : www.picaxe.co.uk . Ce manuel a été préparé en utilisant la version 5.2.0 de Programming Editor. La dernière version est PE6.

La dernière version de ce document est disponible à l'adresse : www.picaxe.co.uk

Si vous avez une question au sujet de n'importe quel commande, n'hésitez pas à poser la question sur les forums (anglais et français) très actifs sur ce site.

Circuits d'Interfaçage Microcontrôleur



Qu'est-ce-qu'un Microcontrôleur PIC ?

Un Microcontrôleur PIC est un simple circuit intégré assez petit pour tenir dans la paume de la main. Les circuits microprocesseur « traditionnels » contiennent quatre ou cinq cuits intégrés séparés – Le microprocesseur lui-même (CPU), une puce mémoire programme EPROM, quelques mémoires RAM et une interface Entrée/Sortie. Avec les microcontrôleurs PIC toutes ces fonctions sont incluses au sein d'un simple boîtier, ce qui les rend rentables et faciles à utiliser.

Les microcontrôleurs PIC peuvent être utilisés comme le « cerveau » pour commander une grande variété de produits. Dans le but de commander des appareils , il est nécessaire de les interfacer (connecter) avec le microcontrôleur PIC.

Qu'est-ce-qu'un Microcontrôleur PICAXE ?

Un microcontrôleur PICAXE est un microcontrôleur standard Microchip PICmicro® qui a été préprogrammé avec le code d'amorçage PICAXE. Le code d'amorçage active le microcontrôleur PICAXE pour être reprogrammé directement via une simple connexion série. Ceci élimine la nécessité d'avoir un programmeur conventionnel (cher), transformant la totalité du système de téléchargement en un simple câble série peu cher !

Le code d'amorçage préprogrammé contient aussi des routines communes (tel que comment générer un temps de pause, ou une sortie son), ainsi chaque téléchargement ne perds pas de temps à télécharger ces données communément requises. Ceci rend le temps de téléchargement très rapide.

Étant donné que les micro-contrôleurs vierges achetés pour « faire » les micro-contrôleurs PICAXE sont achetés en grandes quantités, il est possible au fabriquant de programmer le code d'amorçage et encore vendre le microcontrôleur PICAXE à des prix proches du prix catalogue pour de simples microcontrôleurs PIC non-programmés. Ceci signifie que le coût des micro-contrôleurs PICAXE est peu onéreux pour l'utilisateur final.

Interfaçage du Microcontrôleur PICAXE

Cette section explique comment interfacer plusieurs entrées et sorties d'équipements au micro-contrôleur PICAXE. Les explications des commandes BASIC sont fournies dans la section Commandes (disponible séparément). Les circuits d'interfaçage peuvent aussi être utilisés avec les micro-contrôleurs PIC tel que le PIC16F84A, bien que ces microcontrôleurs peuvent nécessiter la programmation en code assembleur.

Cette section est découpée en quatre sous-sections :

- Introduction aux « circuits d'interfaçage standards »
- Interfaçage Équipement de Sortie
- Interfaçage Équipement d'Entrée
- Interfaçage Avancé de Composants

Remarque sur les Exemples de Code BASIC

Les exemples simples de code BASIC sont fournis au sein de chaque sous-section. Les exemples ne sont pas des exemples « complets » mais des tronçons de code qui ont été introduits au sein d'un programme principal lors de l'utilisation de ce composant particulier. Lors de l'utilisation de ces exemples de codes, il faut se rappeler que :

1. Chaque broche doit être définie comme une entrée ou une sortie avant l'utilisation du code.
2. Si les broches matérielles sont changées par rapport à ce qui est donné par le diagramme du circuit, il est nécessaire de modifier le numéro des broches dans le code.
3. N'importe quelle commande « **let dirs=** » ou « let pins= »ajustera **toutes** les 8 broches dans le port.
4. Essayer de garder les variables indépendantes les unes des autres. Si une sous-procédure utilise une variable, n'utilisez pas la même variable ailleurs dans le code. Si la même variable doit être utilisée encore, assurez-vous qu'il n'y a pas d'incompatibilité avec le reste du code. Ceci est la façon la plus courante d'ajouter des erreurs «difficiles à trouver» dans le code logiciel.

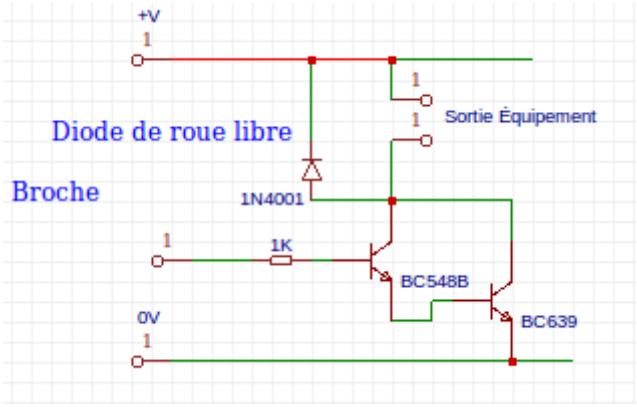
Note sur la Sélection de Composant

Par convenance et pour une compréhension aisée, un simple équipement a été adopté Lors de l'utilisation d'interfaçage standard de composants tels que transistors et MOSFETS. Par exemple, le transistor « standard » sélectionné est le darlington BCX38C. Ceci ne veut pas dire que c'est le seul transistor qui peut être utilisé, mais il est choisi parce qu'il est adapté à la majorité des projets d'applications. Tous les composants listés sont des éléments courants qui peuvent être trouvés chez la plupart des distributeurs d'électronique.

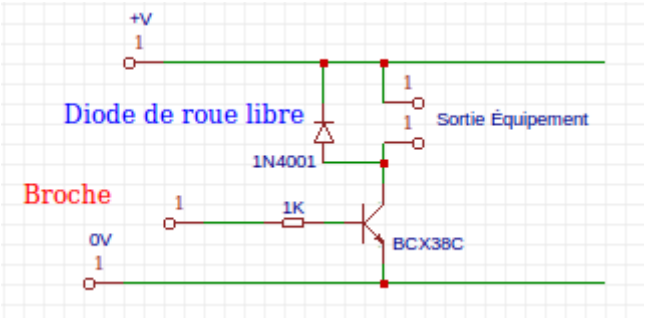
Circuits Standards d'Interfaçage

Circuits Standard 1 – Le circuit d'Interfaçage à Transistor

La plupart des équipements de sortie nécessite une commutation par transistor. Dans la plupart des cas une paire darlington formée de deux transistors est idéale.



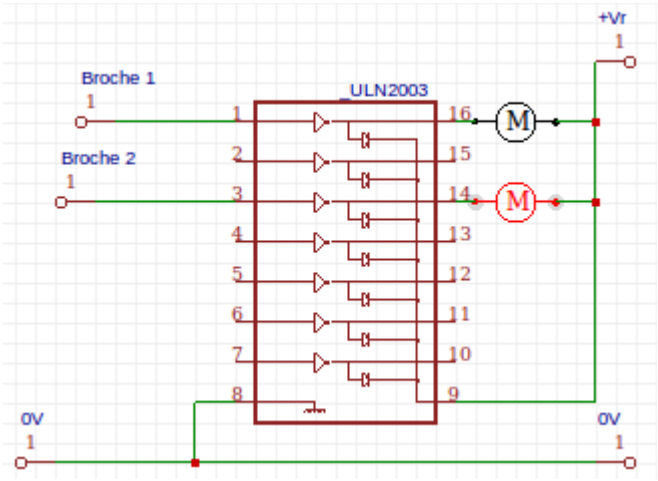
Toutefois ce circuit
demande que deux transistors séparés soient utilisés. Il est possible d'acheter un composant qui comporte les deux transistors dans un seul boîtier. Il s'agit du BCX38C et qui peut couper des courants de 800mA. Ce transistor est utilisé dans tous les montages de ce manuel.



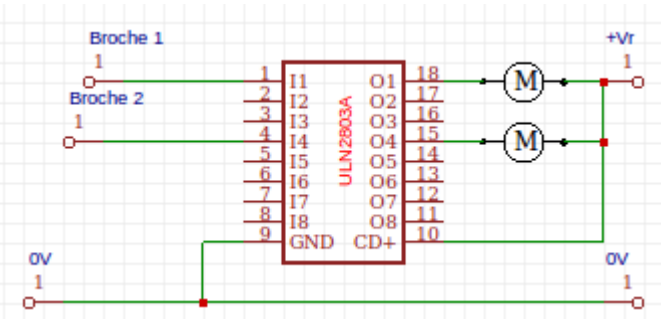
Remarquez qu'il est usuel de connecter une diode dite « de roue libre » aux bornes de la sortie équipement.Elle est essentielle avec les équipements tels que les relais, les solénoïdes et les moteurs qui créent des pic de surtension à la coupure. La diode 1N4001 est le composant recommandé.

Circuits Standards 2 – Utilisation d'un CI Darlington

Si un nombre d'équipements de sorties doit être commandé, il peut être utile d'utiliser des sorties transistorisées. Dans ce cas il est souvent plus facile d'utiliser un circuit driver Darlington ULN 2003. C'est simplement un circuit qui contient 7 transistors darlington semblables au BCX38C. La puce contient aussi les diodes de « roue libre » et les diodes 1N4001 externes ne sont pas nécessaires.



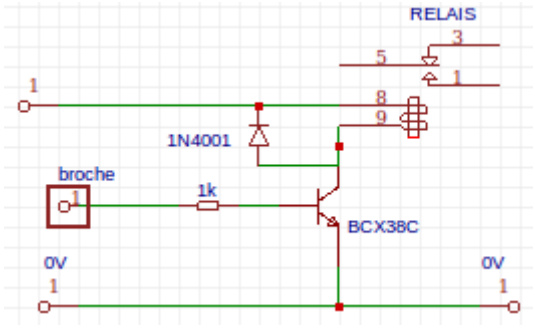
Un composant appelé driver Darlington ULN2803 est aussi disponible. Il est idntique à l'ULN 2003 excepté qu'il comprte 18 broches et contient 8 paires darlingtonau lieu de 7. S'il est nécessaire de passer un courant relativement élevé à travers l'équipement, il peut être pratique d'utiliser une paire de pilote comme dans ce montage.



Le driver Darlington ULN2803 est fourni pré équipé pour les circuits des projet PICAXE.

Circuits Standards 3 – Le Circuit Interfaçage Relais

Un relais peut être utilisé pour couper des équipements de grosse puissance tel que des Realis ou des Solénoïdes. Si désiré, le relais peut être alimenté par une source différente, ainsi, par exemple, 12V, les solénoïdes peuvent être commander par le micro-contrôleur. Remarquez l'utilisation d'une diode de « roue libre »aux bornes du relais. Ceci pour provenir les dommages pouvant subvenir lors de la coupure de la bobine. Une diode 1N4001 convient parfaitement.

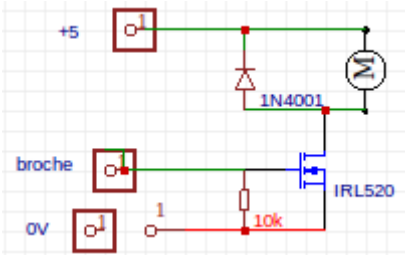


Circuits Standards 4 – Le Circuit Interfaçage à MOSFET

Les MOSFETs de puissance peuvent être utilisés à la place des transistors darlington pour couper des équipements de puissance moyenne. Le circuit standard MOSFET est montré ci-dessous. Le composant IRL520 est un MOSFET a commande par niveau logique.

Remarquez que l'on connecte une diode de « roue libre » aux bornes de l'équipement. Ceci est essentiel avec les équipements tels que relais, solénoïde et moteurs qui créent une étincelle quand l'alimentation est coupée. Une diode 1N4001 convient parfaitement.

Quand une puce PICAXE se réinitialise, la broche de sortie n'est pas pilotée momentanément. Par conséquent sur les circuits sensibles, il peut être nécessaire d'inclure une résistance de tirage vers le bas de 10k sur la grille du MOSFET. Ceci maintient la grille Off jusqu'à ce que le PICAXE pilote activement la sortie.

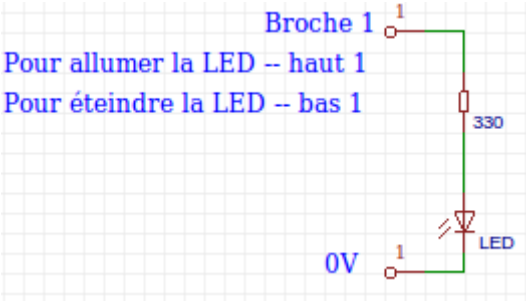


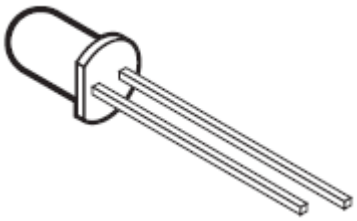
Interfaçage Équipement de Sortie

Équipement de Sortie 1 – LED

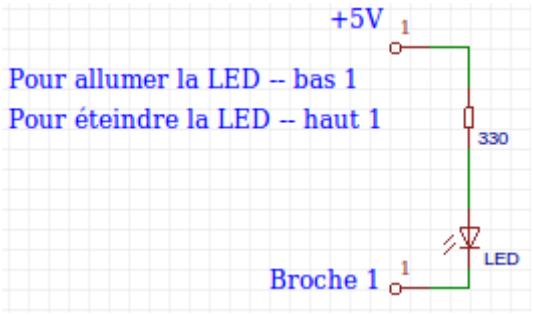
Le micro-contrôleur PICAXE peut absorber ou fournir une petite quantité de courant, ce qui signifie qu'une LED peur être raccordée directement à la broche de sortie. Une résistance série (330 Ω) est aussi requise pour limiter le courant

LED connectée à la masse

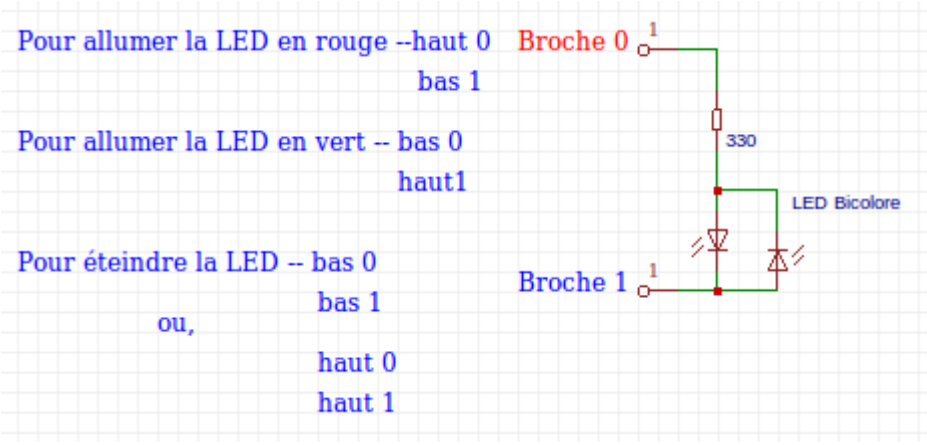




LED connectée au +5V



Les LEDs bicolores contenant souvent des Leds rouge et verte connectées en « inverse parallèle ». Cela signifie que si dans un sens le courant allume la LED verte il allumera la rouge dans l'autre sens. Cependant en utilisant les capacités du micro-contrôleur PICAXE il est possible d'allumer la LED dans les deux couleurs



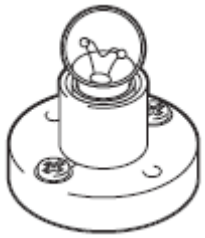
Équipement de Sortie 2 – Voyant

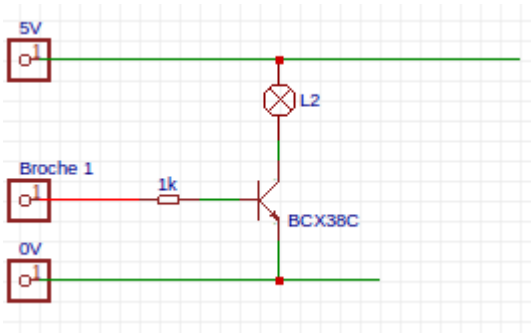
Pour interfacer un voyant, le circuit d'interfaçage à transistor est utilisé. Remarquer que si une source d'alimentation différente pour le voyant est utilisée, les références 0V des alimentations doivent être interconnectées .

Si une batterie est utilisée comme alimentation, il convient de se rappeler qu'une LED consomme moins de courant qu'une lampe. Donc, si une simple indication est requise, une LED sera une bien meilleur solution qu'une lampe, les batteries tiendront plus longtemps.

Pour allumer une Lampe 1 à haut

Pour éteindre la lampe 1 à bas





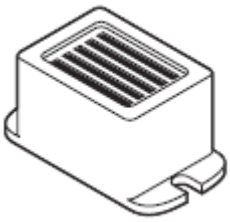
Équipement de Sortie 3 – Buzzer

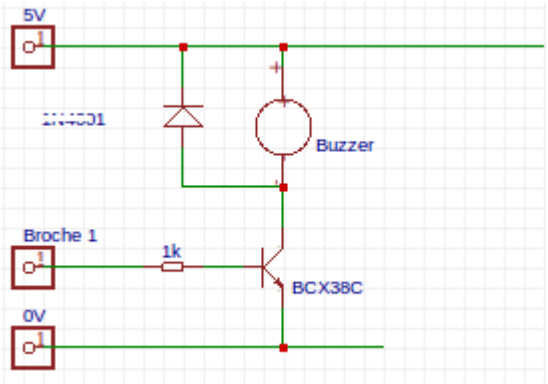
Pour interfacer un buzzer, le circuit d'interfaçage à transistor est utilisé.Noter que si une alimentation différentes est utilisée pour le buzzer, les points 0V des différentes alimentations doivent être connectés pour avoir une référence commune.

Si une batterie est utilisée, il convient de se rappeler que les Piezo acoustiques consomment moins de courant que les buzzers. Les buzzers aussi ne fournissent qu'un ton, là où un piézo acoustique est capable de créer des sons de plusieurs tonalités.

Pour actionner le buzzer 1 à haut

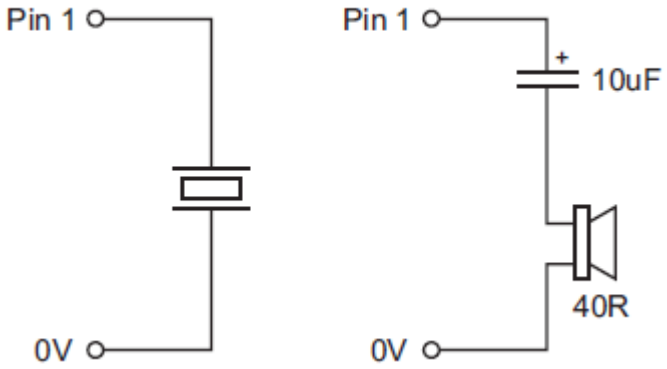
Pour arrêter le buzzer 1 à bas





Équipement de Sortie 4 – Piézo & Haut-Parleur

Un Piézo ou un Haut parleur peuvent être utilisés pour produire des sons variés, où un buzzer ne peut fournir un simple son. Les burzzers produisent un bruit quand l'alimentation est appliquée, mais un pièzo ou un haut parleur requière un signal pulsé pour générer le son. Heureusement, il est très facile de le générer dpuis le micro-contrôleur en utilisant la commande BASIC « sound command ».



Pour produire une note de ton 100, longueur 50 sur la broche 1

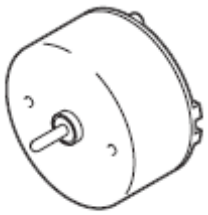
```
sound 1, (100,50)
```

Pour produire un son varié utilisant la variable b1

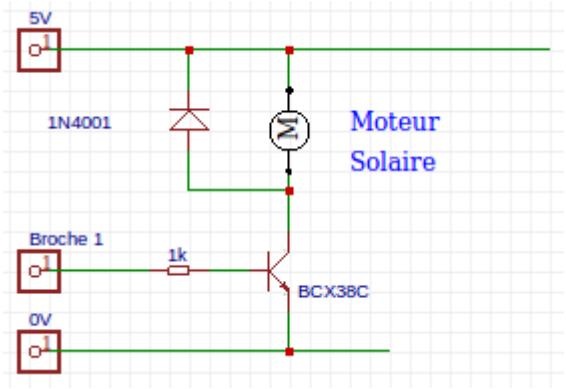
```
for b1 = 1 to 100
    sound, (b1,25)
next b1
```

Équipement de Sortie 4 – Solaire et Moteurs « Jouets » DC

Beaucoup de projets nécessite l'usage de moteur à courant continu pas cher pour créer des mouvements rotatifs. Ces moteurs peuvent être interfacés au microcontrôleur par certain nombre de façons.

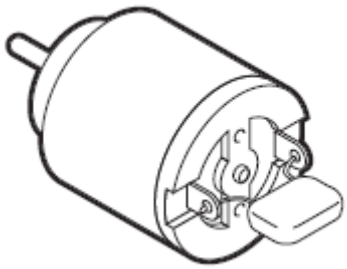


solar motor

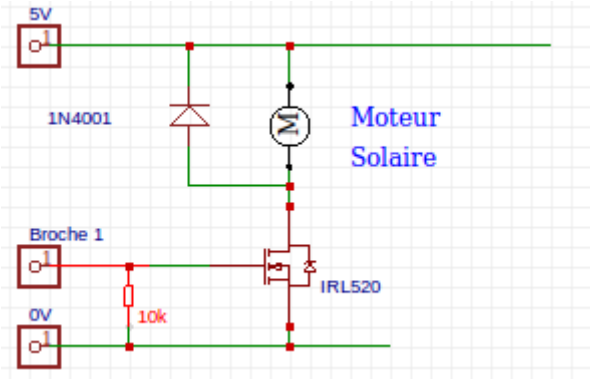


Ce circuit utilise un transistor darlington pour commander le moteur. Ce circuit travaillera avec des moteurs « solaires », mais peu ne pas fonctionner correctement avec des moteur à courant continu pas cher . C'est parce que ce type de moteur introduit un lote de bruit électrique la ligne d'alimentation. Ce bruit peut affecter le micro-contrôleur, et dans quelques cas stopper le complètement le fonctionnement du programme de commande.

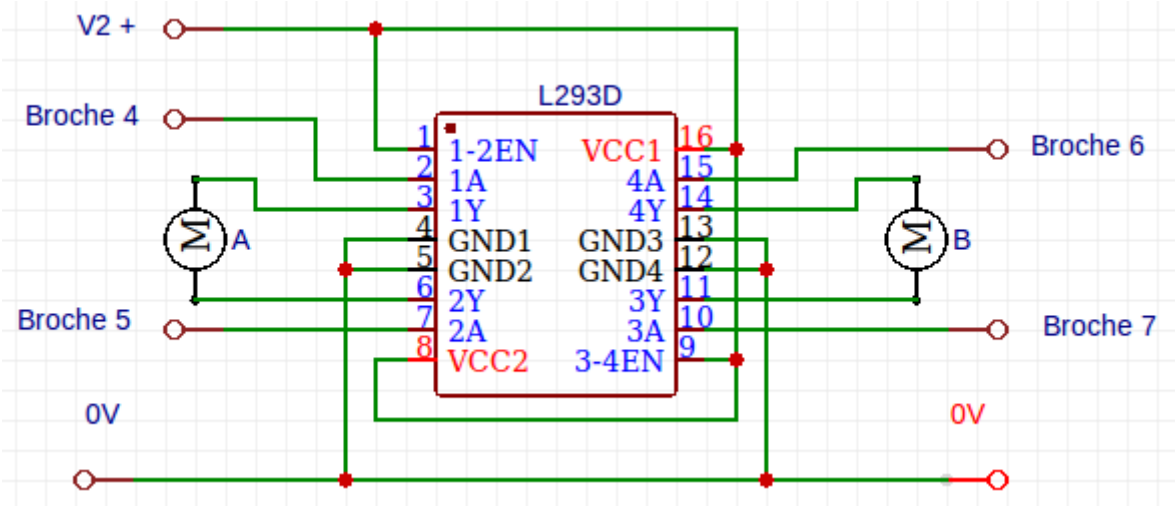
Les bruits électriques peuvent être réduits par l'ajout d'un condensateur soudé aux bornes du moteur. Comme montré ci-contre .
Utilisez un condensateur polyester de 220nF (non polarisé).



Dans l'optique de commander des moteurs de puissance moyenne, un MOSFET de puissance est utilisé à la place du transistor Darlington. Le circuit MOSFEET est montré ci-dessous. Le composant est un IRL 520 de puissance appropriée pour ce type de montage.

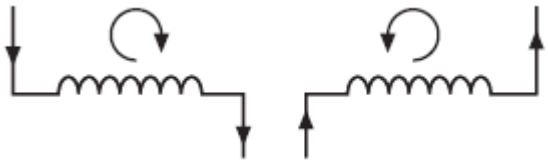


Dans plusieurs occasions, il peut être nécessaire de contrôler deux moteurs. Une approche adéquate et peu onéreuse serait d'utiliser un CI pilote de moteur tel que le LD293D. Ce circuit permet de commander deux moteurs, par l'utilisation de quatre lignes de données depuis le micro-contrôleur. Naturellement, si un seul moteur doit être commandé alors seulement deux lignes seront utilisées.



Deux entrées Bas	- arrêt moteur
1° sortie Haut, 2 ième sortie Bas	- moteur avant
1° sortie Bas, 2 ième sortie Haut	- moteur inverse
2 sorties Haut	- arrêt moteur

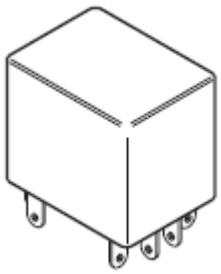
Le changement de l'état des broches d'entrées a pour effet de modifier la direction du flux du courant travers le moteur, comme montré ci-dessous.



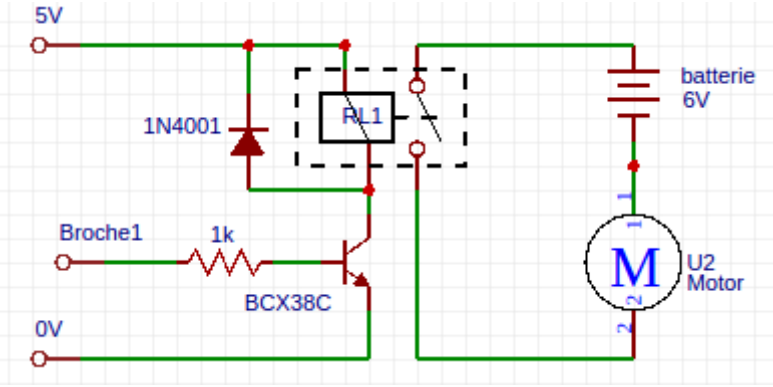
Flux du courant

Noter que le L293D chauffe en usage continu. Un radiateur sur le dessus de la puce l'aidera à refroidir.

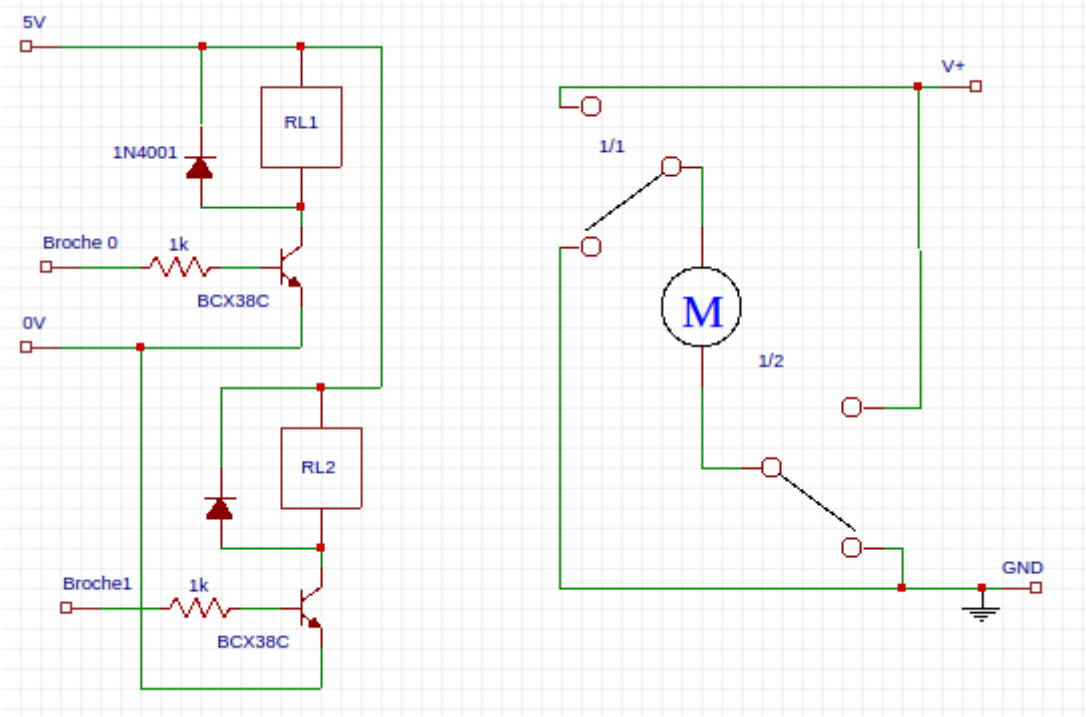
Une solution pour éviter que le bruit électrique n'affecte le micro-contrôleur est d'utiliser des alimentations séparées pour les « commandes » électroniques et le moteur.Par exemple une batterie PP3 doit être choisie pour alimenter le micro-contrôleur et une cellule de 4xAA pour alimenter les moteurs. Naturellement il est nécessaire de lier les deux circuits pour que le moteur puisse être commander. Un relais est un composant idéal pour le faire.



relais



Le circuit ci dessus ne fait que couper le moteur. Si le moteur doit tourner dans les deux sens, deux relais peuvent être utilisés comme montré ci-dessous.



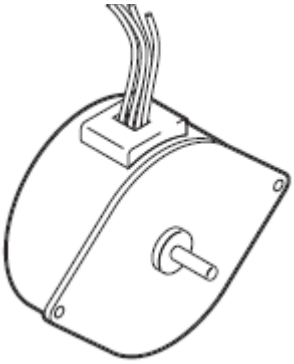
Ce schéma est une version corrigée.
Dans la version d'origine il semblerait qu'il y est une erreur concernant le contact 2/1, il ne faut pas l'utiliser, il n'autorise pas le fonctionnement du moteur dans le commande par la broche 1.

Équipement de Sortie 6 – Moteur pas-à-pas Unipolaire

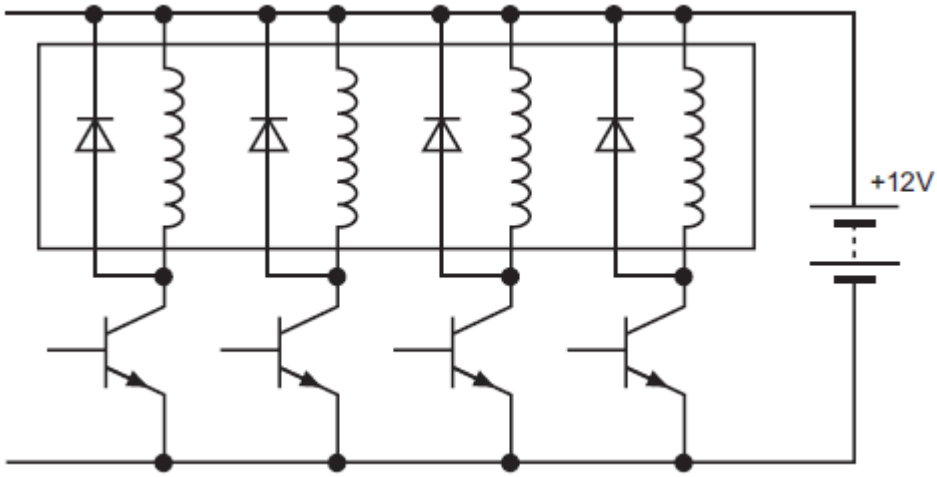
Les moteurs sont des moteurs pas à pas très précis qui sont couramment utilisés dans le disque d'ordinateurs, les imprimantes et horloges. Contrairement aux moteurs à courant continu, qui tournent continuellement lorsque l'alimentation est appliquée, le moteurs pas à pas à besoin que son alimentation soit pulsée en continu avec des caractéristiques spécifiques. Pour chaque impulsion, le moteur pas à pas se déplace autour d'un «pas», souvent 7,5 degrés (donnant 48 pas pour un tour complet).

Il y a deux sortes de moteurs pas-à-pas – Unipolaire et Bipolaire. Les moteurs Unipolaires ont habituellement quatre bobinages qui sont commutés à l'aide d'une séquence particulière. Les moteurs Bipolaires ont deux bobinages dans lesquels le courant est inversé avec une séquence similaire. L'utilisation des moteurs bipolaire est expliquée dans la section suivante.

Chaqu'un des quatre bobinage dans un moteur pas-à-pas unipolaire doit être commuté On et Off dans un certain ordre pour faire tourner le moteur. La plupart des systèmes à microprocesseur utilisent quatre lignes de sortie pour le commander, chaque ligne de sortie commandant l'alimentation d'une des bobines



moteur pas-à pas

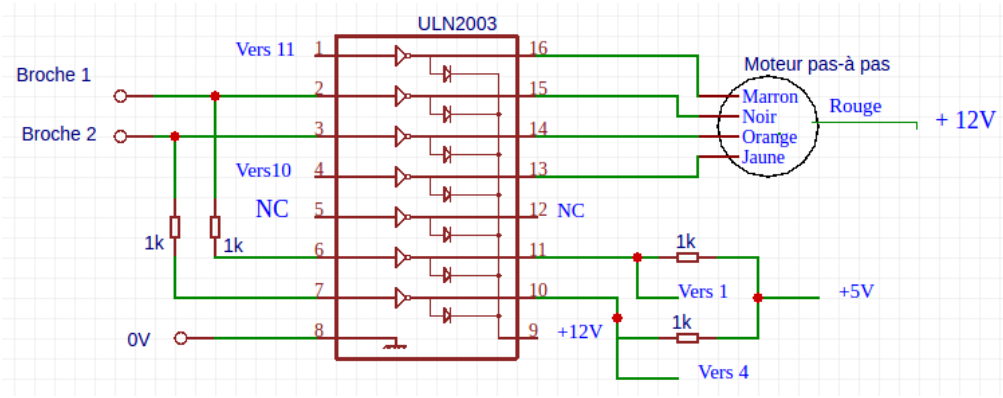


Comme le moteur pas-à-pas fonctionne en 12V, le circuit standard à transistor est utilisé pour commuter chaque bobine. Comme la bobine crée une tension la coupure, une diode de roue libre est nécessaire sur chaque bobine. Le tableau ci-dessous montre les quatre étapes nécessaire pour faire tourner le moteur.

Step	Coil 1	Coil 2	Coil 3	Coil 4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

Observez attentivement le tableau, et notez qu'un schéma est est visible. La bobine 2 est toujours l'opposée (ou logique NOT) de la bobine 1. la même chose s'applique aux bobines 3 et 4. Il est donc possible de réduire le nombre de broches nécessaire du micro-contrôleur à deux seulement par l'utilisation de deux autres portes NON.

Heureusement le driver darlington IC ULN2003 peut être utilisé pour fournir à la fois la porte NOT et le driver darlington. Il contient aussi les diodes de « roue libre » ainsi aucune diode externe n'est nécessaire. Le circuit complet est montré ci-dessous.



N.B. les couleurs des fils du moteur peuvent être différentes

Avant la programmation, il y a un autre modèle pour noter les séquences. Regarder ce tableau, qui montre juste les bobine 1 et 3

Step	Coil 1	Coil 3	Change
1	1	1	
			coil 3
2	1	0	
			coil 1
3	0	0	
			coil 3
4	0	1	
			coil 1
1	1	1	

Noter le changement de l'étape 1à 2, il n'y a que la bobine 3 qui change, Regarder alors le changement suivant – juste la bobine 1 change. En fait les deux bobines changent«à tour de rôle» Haut à Bas. Ce changement Haut -Bas- Haut peut-être décrit comme un état de bascule. Ceci facilite la programmation par l'usage de la commande BASIC `toggle` .

```
steps :     toggle1          ' Bascule broche 1
            pause 200        ' Attendre 200ms
            toggle2          ' Bascule broche 2
            pause 200        ' Attendre 200ms
            goto steps       ' Boucle
```

Remarque :Si le moteur a tendance à être instable , essayer d'ajuster la polarité des fils

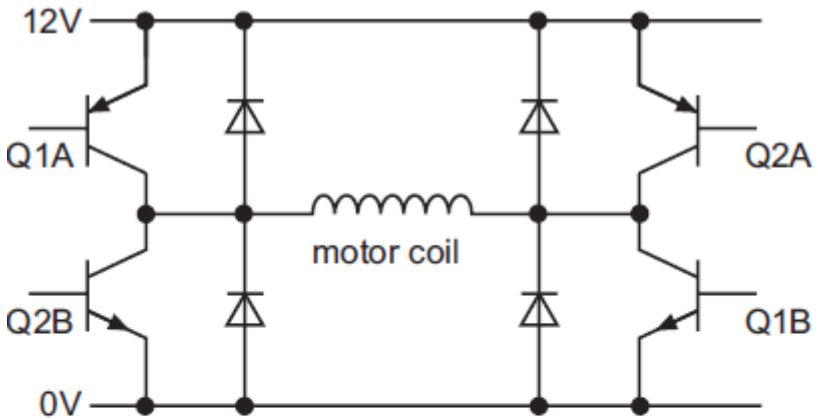
Équipement de Sortie 7 – Moteur pas-à-pas Bipolaire

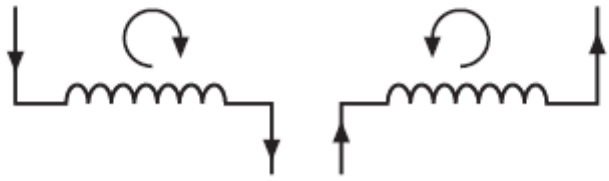
Les moteurs pas à pas sont des moteurs très précis qui sont couramment utilisés dans les disques durs d'ordinateurs, les imprimantes et les horloges. Contrairement aux moteurs à courant continu, qui tournent librement lorsque l'alimentation est appliquée, les moteurs pas-à-pas ont besoin d'une alimentation pulsée permanente suivant une spécification particulière. A chaque pulsation, le moteur avance d'un « pas » souvent 7,5 degrés (ce qui donne 48 pas pour un tour complet) .

Il y a deux types principaux de moteurs – Unipolaire et Bipolaire. Les moteurs Unipolaires ont habituellement quatre bobinages qui sont commutés à l'aide d'une séquence particulière. Les moteurs Bipolaires ont deux bobinages dans lesquels le courant est inversé avec une séquence similaire. L'utilisation des moteurs unipolaire est expliquée dans la section précédente.

Le moteur bipolaire a deux bobinages qui peuvent être commandés, ainsi que le sens du courant dans les différentes directions à travers les bobines dans un certain ordre. Les changement de champs magnétique créé par ces bobines provoquent la rotation du moteur par « pas ».

Le circuit qui est normalement utilisé pour commander une des bobines est montré- ci dessous. Remarquer qu'il y a quatre transistors de commande, qui sont commander par « paires ». Donc pour deux bobines, il y a quatre paires de transistors (Q1 -Q4) qui sont commutées suivant une séquence particulière.





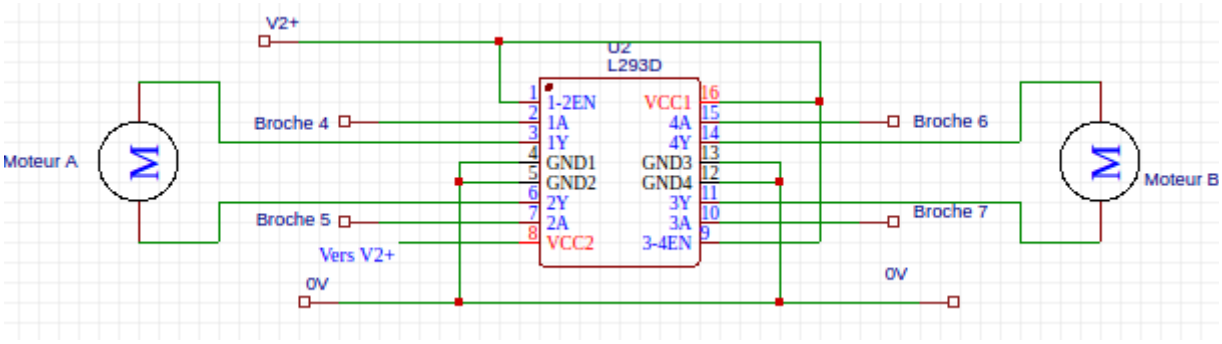
Current flow

Noter que comme les bobines produisent une surtension à la coupure, 8 diodes de « roue libre » sont nécessaires.

La table ci-dessous montre les quatre étapes différentes pour faire tourner le moteur :

Step	Q1	Q2	Q3	Q4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

Heureusement le pilote de moteur L293D a été spécialement fabriqué pour fournir ces transistors de commutation. Le L293D contient les 8 transistors et les diodes au sein d'un circuit 16 broches.

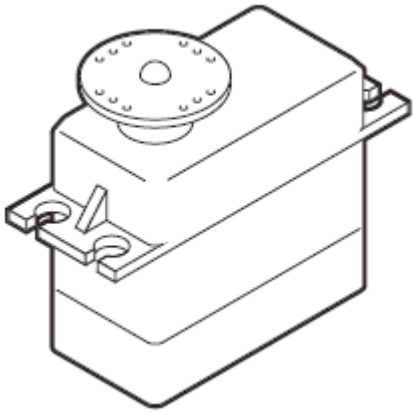


Les quatre broches du micro-contrôleur sont reliées au quatre « paires » par les broches 2, 7, 10 et 15 du L293D

Cette exemple de procédure fait tourner le moteur de 100 pas vers la gauche et 100 pas vers la droite en utilisant deux sous-procédures. `lstep` fait tourner le moteur d'un pas vers la gauche, `rstep` idem mais vers la droite. La variable `b1` est utilisée pour stocker la position du pas et ne doit pas être utilisée ailleurs dans le programme.

```
main : for b3 = 0 to 99                          '  lance une boucle for…..next
                gosub lstep                      '  appelle la sous-procédure pas à gauche
       next b3                                   '  boucle suivante
       for b3 = 0 to 99                          ' lance une boucle for…..next
                gosub rstep                      ' lance la sous-procédure pas à droite
       next b3                                   ' boucle suivante

lstep : let b1 =b1 + 1                           ' ajoute 1 à la variable b1
       goto step2                                ' aller à  la table de conversion
rstep :  let b1 = b1 – 1                         ' soustrait 1 de la variable b1
       step2 : let b1 = b1 & %00000011           ' masque les 2 bits de poids faible de b1
       lookup b1, (%1010,%1001,%0101,%0110), b2  ' code convertit dans b2
       lets pins = b2                            ' b2 sortie sur la ligne de commande
       return
```

Équipement de Sortie 8 – Servo Radio Commande



Les servos sont utilisés dans la plupart des avions, des bateaux et des voitures radiocommandés pour commander les systèmes de pilotages. Ils ont des équipements précis qui pivote du même angle pour un signal donné, et et ainsi sont idéals pour une utilisation dans des machines automatisées.

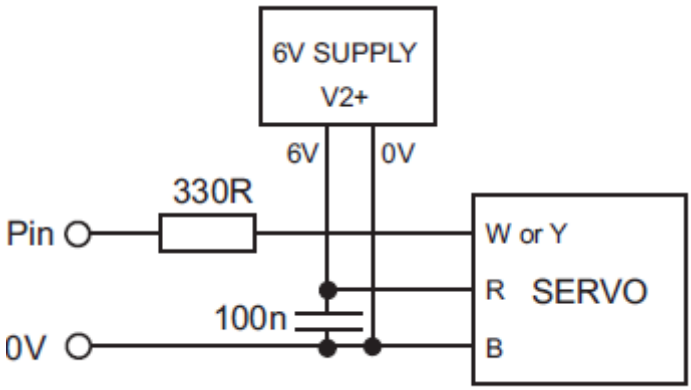
Les servos peuvent être pilotés directement via la commande `servo` ou par des commandes `pulsout`.

Un servo type a 3 fils de connexion, normalement rouge, noir, blanc (ou jaune). Le fil rouge est l'alimentation 5V, le fil noir est le 0V de l'alimentation et le blanc (ou jaune) est pour le signal de position.

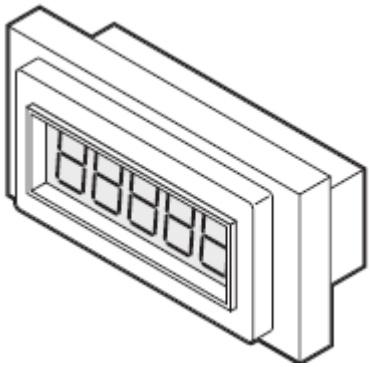
Le signal de position est une impulsion comprise entre 0,75 et 2,25 millisecondes (ms) de long, répétées environ toutes les 18 ms (ainsi il y a environ 50 impulsions par seconde). Avec une impulsion de 0,75 ms le servo tourne vers une de ses butées et avec une une impulsion de 2,25 ms il tourne vers l'autre butée. Donc avec une impulsion de 1,5 ms, le servo viendra en position centrale. Si les impulsions sont arrêtées, le servo tournera librement vers n'importe quelle position.

Malheureusement les servos demande un courant important (au moins 1 A) et ainsi introduisent un bruit important dans la ligne d'alimentation. Donc dans la plupart des cas le servo devra être alimenté par une alimentation séparée, comme montré ci-dessous. Rappelez-vous que lors de l'utilisation de deux alimentations séparées, les deux 0V doivent être connectés ensemble pour fournir une référence commune.

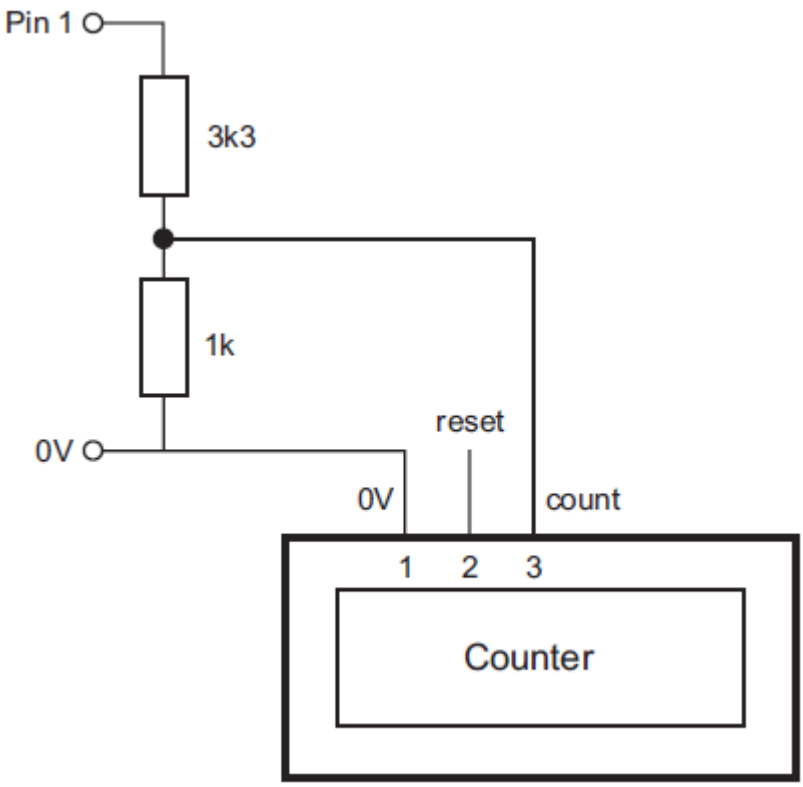
```
  init :  servo 4, 75                ' démarre le servo sur  broche 4
main :  servopos 4 , 75              ' déplace le servo à une de ses butées
        pause 2000                   ' attend 2 secondes
        servopos 4 , 150             ' déplace le servo au centre
        pause 2000                   ' attend 2 secondes
        servopos 4 , 225             ' déplace le servo vers l'autre butée
        goto main                    ' revient au départ
```



Équipement de Sortie 9 – Module Compteur



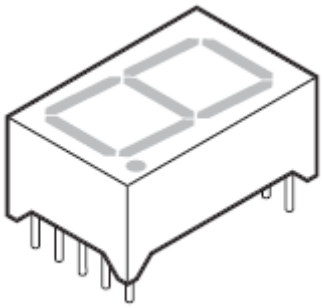
Le Module Compteur est un afficheur numérique LCD qui peut être utilisé pour montré une valeur de compteur. Pour incrémenter le compteur une impulsion (entre 1 et 1,5 v) doit être appliquée au compteur sur la borne 3 . Comme le micro-contrôleur PICAXE fonctionne avec du 5v, un diviseur de potentiel formé de résistances doit être utilisé pour réduire la tension du signal de sortie du micro-contrôleur PICAXE à 1,5V Comme le compteur utilise sa propre batterie 1,5 V interne, les deux 0V doivent être interconnectés.



Pour incrémenter le compteur : pulsout 1 , 100

Pour réinitialiser le compteur, un second diviseur de tension est ajouté et connecté à la borne 2.

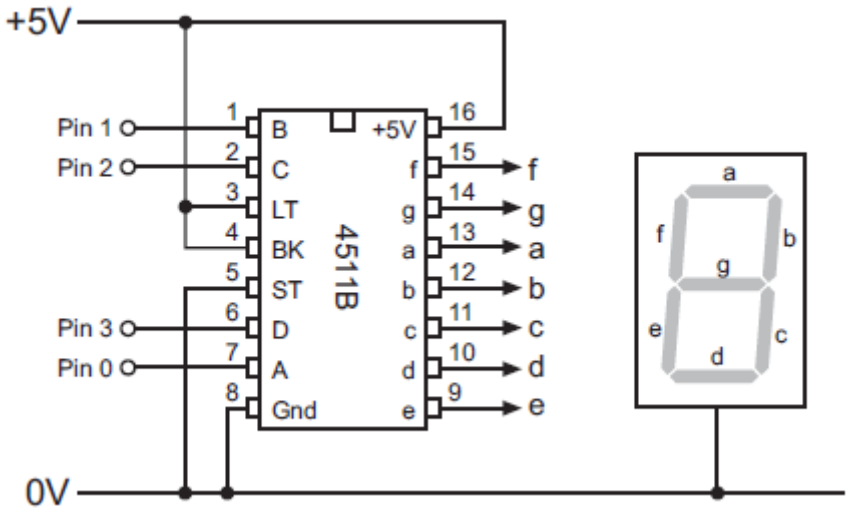
Équipement de Sortie 10 – Afficheur Sept Segments



Un afficheur 7 segments contient 7 barres LED qui peuvent être allumées suivants différentes combinaisons pour montrer les 10 digits de 0 à 9. En théorie chaque barre peut être connectée à une broche de sortie d'un micro-contrôleur, mais cela occuperait 7 des 8 sorties disponibles.

Une meilleur solution est d'utiliser un circuit intégré dédié tel que CMOS 4511B pour commander l'affichage sept segments. Ce CI commande l'affichage sept segments en accord avec le « code » binaire sur les quatre lignes de donnée. Ce système utilise 4 broches au lieu de 7.

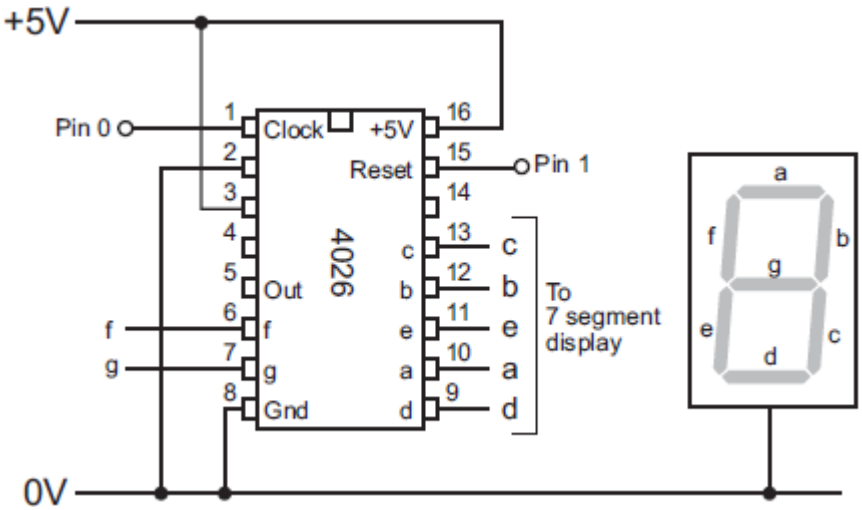
NOTE IMPORTANTE – L'affichage sept segments est disponible en deux type appelés « cathode commune » et « anode commune ». Les circuits suivant travaille seulement avec l'afficheur sept segments du type « cathode commune ». Utilisez les datasheet des fabricants pour déterminer la disposition des broches de sortiedes barres de LED.



Cet exemple de code compte les digits de 0 à 9 :

```
main :  for b1 = 0 to 9          ' définition d'une boucle for…..next  utilisant la variable b1
        let pins =b1             ' Sortie de b1  sur les quatre lignes de donnée
        pause 1000               ' pause 1 seconde
        next b1                  ' suivant
        goto main                ' boucle vers le départ
```

Une autre solution possible est l'utilisation du CMOS 4026B pour commander l'afficheur sept segments. Ce système utilise juste deux broches pour commander l'afficheur. La broche réinitialisation est utilisée pour remettre l'afficheur à 0. la broche horloge est alors utilisée pour incrémenter le chiffre à partir de 0. Ceci signifie que pour afficher le chiffre « 4 » il est nécessaire de réinitialiser et ensuite de pulser la ligne de l'horloge 4 fois. En réalité ceci signifie que l'afficheur montre les chiffres 0- 1- 2- 3- 4, mais comme ils sont activés extrêmement rapidement, l'œil humain ne peut pas voir les changements et ainsi le chiffre « 4 » semble apparaître immédiatement.



Cet exemple de code utilise la sous- procédure « clock » pour afficher le chiffre « 4 » qui est chargé dans la variable b1

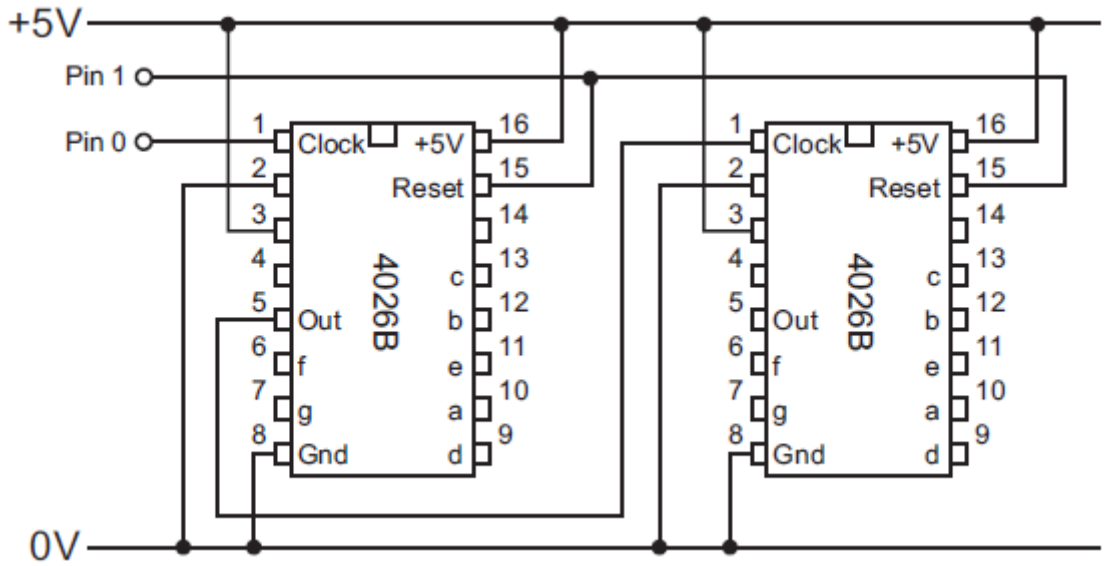
Ceci est la sous-procédure :

```
clock :     pulsout 1, 10           ' réinitialisation afficheur à 0
            if b1 = 0 then endclk   ' si b1 = 0 alors retour
            for b3 = 1 to b1        ' démarre  une boucle for...next
            pulsout 0,10            ' impulsion sur ligne horloge
            next b3                 ' boucle suivante
            encclk : return         ' retour depuis la sous-procédure
```

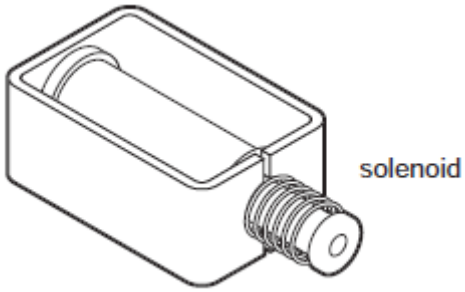
Ceci est le programme principal :

```
main :      let b1 = 4              ' donne à la variable b1 la valeur 4
            gosub clock             ' appelle la sous-procédure
            pause 1000              ' attends 1 seconde
            goto main               ' boucle
```

Ce système peut être étendu à deux chiffres en ajoutant un deuxième 4026 et un second afficheur sept segments, comme montré dans le schéma ci-dessous. Pas de changement de code nécessaire, mais juste donner à la variable b1 une valeur entre 0 et 99 et le nombre qui devra être affiché sue les deux afficheurs quand la sous-procédure « clock » est appelée.



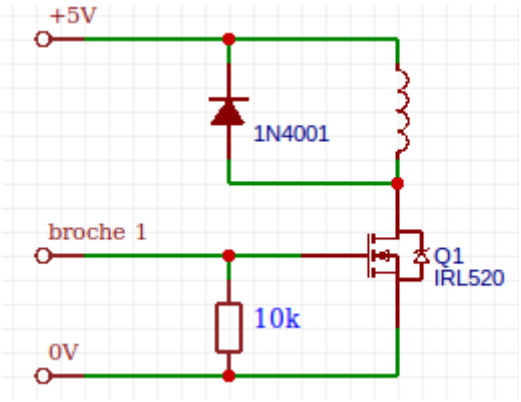
Équipement de Sortie 11 – Solénoïde & Électrovanne



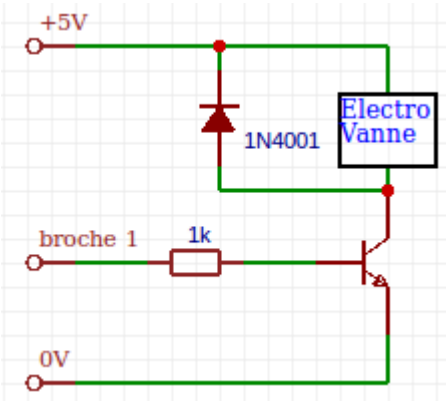
Un solénoïde est constitué d'un noyau
plongeur en acier à l'intérieur d'une bobine électrique qui est bobinée autour d'un tube. Quand la bobine est alimentée un champs électrique est créé, et cela tire le piston dans le tube. Quand la

bobine est désexcitée un ressort pousse le piston en dehors du tube.

Pour commander un solénoïde le circuit MOSFET standard est utilisé :



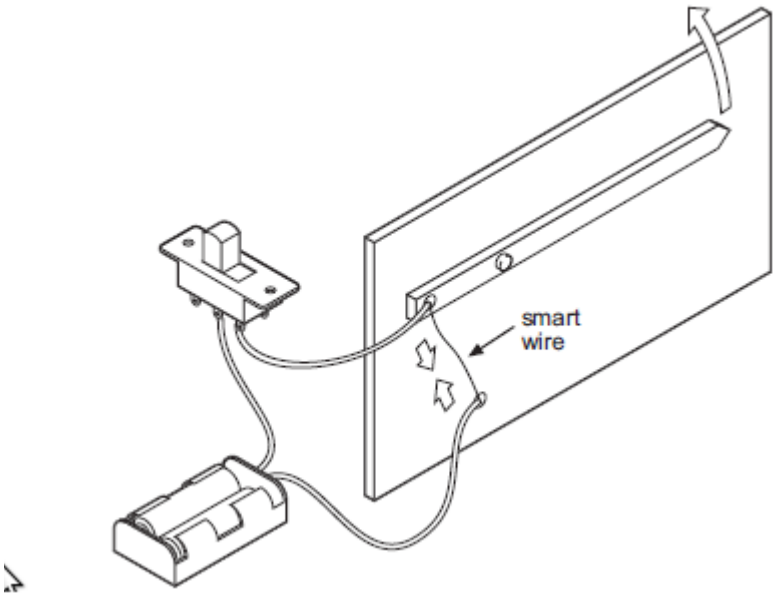
L'électrovanne Isonic peut être utilisée pour commander un flux d'air à travers un système pneumatique. Les valves Isonic sont idéales pour les batteries utilisées en production car elles travaillent en basse tension et tirent moins de courant que les électrovannes traditionnelles. Le circuit de commutation à transistor peut être utilisé pour piloter des vannes Isonic



Pour alimenter la bobine broche 1 Haut

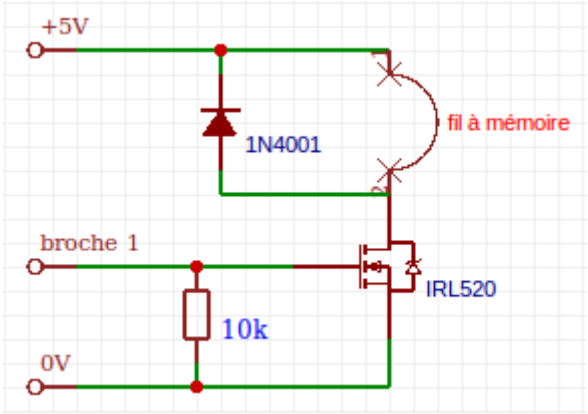
Pour couper la bobine broche 1 Bas

Équipement de Sortie 12 – Fil à Mémoire



Les fils ou ressorts à mémoire de forme sont des matériaux « intelligents » qui peuvent être utilisés pour créer un mouvement mécanique. Quand un courant électrique traverse le fil, celui-ci chauffe et se contracte avec une grande force de traction. Quand le courant est coupé, le fil refroidit et ainsi se dilate (un ressort en acier «traditionnel» est parfois utilisé pour tirer la fil / ressort tendu comme il se refroidit).

Le fil/ressort intelligent demande un courant relativement important, ainsi le circuit d'interfaçage standard à FET peut être utilisé pour interfacer avec le microcontrôleur.



Pour contracter le fil/ressort broche 1 Haut

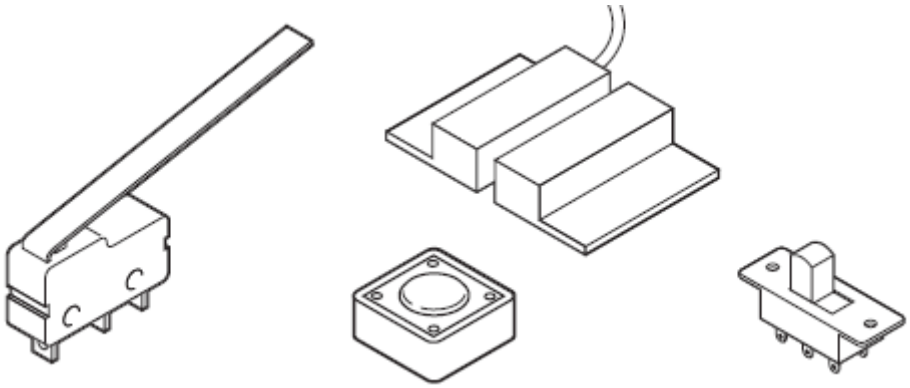
Pour le laisser se dilater broche 1 Bas

Remarque : ne pas oublier que le fil à mémoire demande une régulation en courant pour contrôler son cycle contraction/dilatation sans être détruit.

Interfaçage Équipement d'Entrée

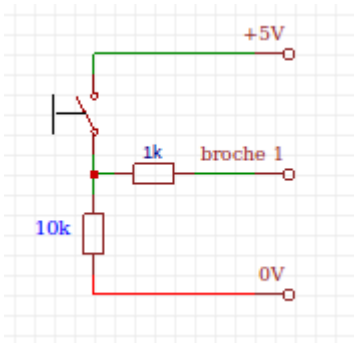
Équipement d'Entrée 1 – Interrupteurs

Il y a une grande variété de commutateurs disponibles, mais la la majorité ont deux contacts qui sont soit ouverts Off) soit fermés (On). Les deux circuits montrés ci-dessous peuvent être utilisés avec presque tous les interrupteurs.



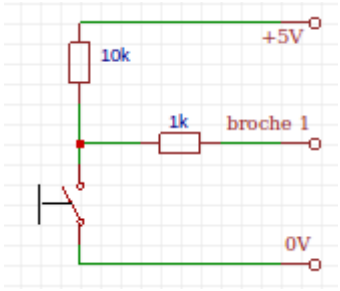
Avec ce circuit la broche d'entrée est Bas quand l'interrupteur est ouvert et Haut quand celui-ci est fermé.

Goto « jump » quand l'interrupteur est ouvert : if pin0 = 0 then jump
Goto « jump » quand l'interrupteur est fermé : if pin0 = 1 then jump



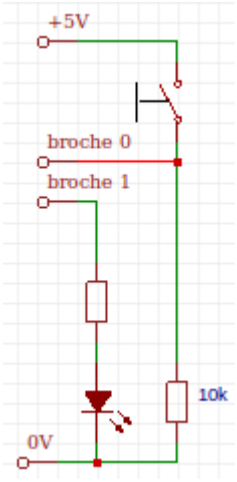
Avec ce circuit la broche d'entrée est Haut quand l'interrupteur est ouvert et Bas quand celui-ci est fermé.

Goto « jump » quand l'interrupteur est ouvert : if pin0 = 0 then jump
Goto « jump » quand l'interrupteur est fermé : if pin0 = 1 then jump



Interrupteurs à Rebonds

Tous les interrupteurs mécaniques « rebondissent » à l'ouverture ou à la fermeture. Ceci signifie que le contact de l'interrupteur « rebondit » ouvert/fermer avant de se fixer. Bien que le micro-contrôleur PICAXE agisse aussi rapidement que possible, il est possible que dans plusieurs programme le micro-contrôleur enregistre 2 ou 3 de ces « rebonds » au lieu d' enregistrer une seule fermeture.



Le plus simple pour supprimer le rebond d'un circuit est de simplement ajouter une temporisation (pause 100) après la commande if …. . Si la section de code après l'action sur le poussoir est assez longue cette temporisation se produit naturellement (pendant que les autres commandes de code sont effectuées) et ainsi cette temporisation n 'est pas nécessaire. Donc si le code n'est pas long, comme dans l'exemple suivant, une commande `pause` être utilisée à la place.

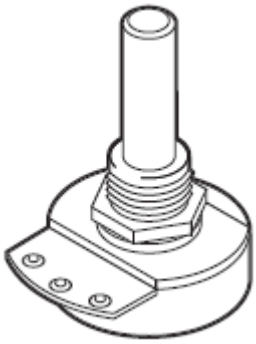
Les deux programmes suivants montre l'effet d'interrupteur rebondissant. Le programme devrait allumer une LED sur la broche 1 quand l'interrupteur connectée à la broche 0 a été pressé plus de 5 fois. Toutefois le premier comptage peut ne pas fonctionner correctement, parce que le micro-contrôleur peut compter des « rebonds » avant l'action actuelle, et ainsi la LED peut s 'allumer prématurément.

```
init : let b0 = 0

main : if pin 1 = 1 then add
       goto main

add :  letb0 = b0 + 1
       if b0 < 5 then main
       high 1
       goto main
```

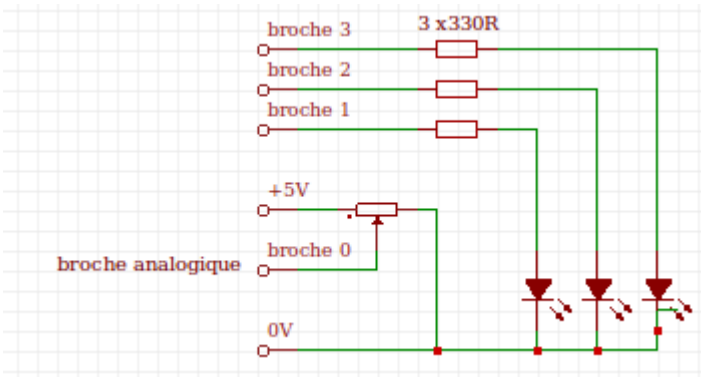
```
init : let b0 = 0

main : if pin 1 = 1 then add
       goto main

add : pause 100              ' délai court
      let b0 =  b0 + 1
      if b0 < 5 then main
      high 1
      goto main
```

Équipement d'Entrée 2 – Potentiomètre

Un potentiomètre (ou « résistance variable ») a un curseur qui peut être déplacé pour changer la valeur de la résistance du potentiomètre. Ceci peut être utilisé pour mesurer un mouvement rotatif ou linéaire.



La commande `readADC` est utilisée pour mesurer la valeur de la résistance en effectuant une conversion analogique-numérique.. Cette valeur de la résistance est donnée dans une fourchette de 0 à 255 et est stockée dans une variable. Après stockage de la lecture dans la variable la commande `if….then` peut être utilisée pour réaliser différentes fonctions

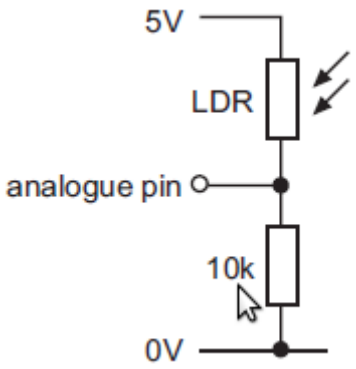


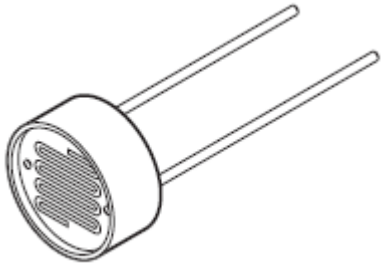
Le programme ci-dessous allume les trois différentes LEDs (connectées aux broches 1, 2 et 3) selon la lecture du capteur analogique

```
main :  readadc 0,b1          ' lire la valeur sur la broche 0 dans la variable b1
        if b1< 75 then light1 ' si b1 est < 75 alors light 1
        if b1< 175 then light2 ' si b1 est < 175 alors light 2
        goto light3           ' si b1 est > 175 alors light 3

light1 : high 1               ' allumer LED 1
        low 2                 ' éteindre LED2
        low 3                 ' éteindre LED 3
        goto main             ' boucle

light2 : low 1                ' éteindre LED 1
        high 2                ' allumer LED2
        low 3                 ' éteindre LED 3
        goto main             ' boucle

light3 : low 1                ' éteindre LED 1
        low 2                 ' éteindre LED2
        high 3                ' allumer LED 3
        goto main             ' boucle
```

Équipement d'Entrée 3 – Varistance (LDR)

Une résistance dépendant de la lumière (LDR) est une résistance de valeur en fonction de la lumière qui l'éclaire. Un composant couramment utilisé, le ORP -12 a une forte résistance dans l'obscurité et une résistance faible dans la lumière. La connexion du LDR au micro-contrôleur est très simple, mais certains logiciels d'étalonnage sont nécessaires.



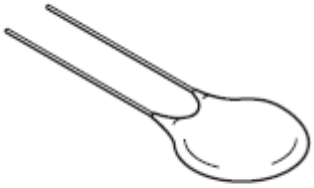


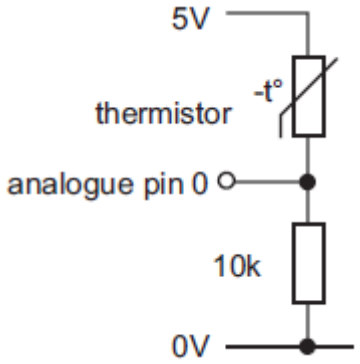
Il faut se rappeler que la réponse d'une LDR n'est pas linéaire, aussi les lectures ne changeront pas exactement de la même façon qu'avec un potentiomètre linéaire. En général, il y a un changement de résistance plus grande vers des niveaux de lumière plus brillants. Cela peut être compensé dans le logiciel en utilisant une fourchette plus petite à des niveaux de lumière plus sombres. Faites des essais pour trouver les réglages les plus appropriés pour le circuit.

```
main :  readadc 0,b1            ' lire la valeur
        if b1 <50 then light1   ' fourchette 0-50 =50
        if b1<100 then light2   ' fourchette 50-100 = 50
        if b1<145 then light3   ' fourchette 100-145 =45
        if b1<175 then light4   ' fourchette 145-175 =30
        goto main
```

Équipement d'Entrée 4 – Thermistance

Une thermistance est une résistance dont la valeur change en fonction de la température. En réalité, toutes les résistances changent de valeur car ils chauffent ou se refroidissent, mais les thermistances sont fabriquées pour montrer un grand changement de résistance. Le raccordement de la thermistance au microcontrôleur est très simple, mais un logiciels d'«étalonnage» peut être nécessaire..





Il faudra se rappeler que la réponse d'une thermistance n'est pas linéaire et ainsi les lectures ne changeront pas de la même manière qu'avec un potentiomètre linéaire. En général il y a un grand changement de résistance aux températures basses. Cela peut être compensé par un logiciel en utilisant une fourchette plus petite aux températures élevées. Il faudra expérimenter pour trouver les réglages les plus appropriés pour le circuit.

```
main :  readadc 0,b1              ' lire la valeur
        if b1 <50 then light1     ' fourchette 0-50 =50
        if b1<100 then light2     ' fourchette 50-100 = 50
        if b1<145 then light3     ' fourchette 100-145 =45
        if b1<175 then light4     ' fourchette 145-175 =30
        goto main
```

Interfaçage Composants Avancés

Interfaçage Avancé 1 – Affichage LCD

Un Afficheur à Cristaux Liquide (LCD) est un composant électronique qui peut être utilisé pour montrer du texte ou des chiffres. Il y a deux types principaux d'afficheurs LCD afficheurs numériques (utilisé dans les montres, les calculatrices) et les afficheurs texte alphanumériques (souvent utilisés dans les photocopieurs et les téléphones mobiles).

L'afficheur est fait d' un certain nombre de forme de « cristaux ». Dans les afficheurs numériques ces cristaux sont formés de « barres » et dans les afficheurs alphanumériques les cristaux sont simplement disposés suivant des modèles de « points ». chaque cristal a une connexion électrique individuelle, ainsi chaque cristal peut être commandé indépendamment. Quand le cristal est « Off » (quand aucun courant ne le traverse) le cristal reflète la même quantité de lumière que le matériau de base, et ainsi les cristaux ne peuvent pas être vus. Cependant quand le cristal est parcouru par un courant, il change de forme et ainsi absorbe plus de lumière. Ceci fait apparaître le cristal plus sombre à l'œil humain – et ainsi la forme du point ou de la barre peut être vue sur le fond.

Il est important de réaliser la différence entre un afficheur LCD et un afficheur à LED. Un afficheur à LED(souvent utilisé dans les radio-réveils) est d'un nombre de LEDs qui donnent effectivement de la lumière (et donc on peut le voir dans l'obscurité). Un écran LCD ne réfléchit pas la lumière, et ne peut donc pas être vu dans l'obscurité.

Caractères LCD

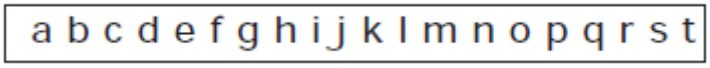
La table de la page suivante montre les caractères disponibles depuis un afficheur LCD typique. Le « code » caractère est obtenu en ajoutant le nombre en haut de la colonne au nombre sur le coté de la rangée

Notez que les caractères 32 à 127 sont toujours les mêmes pour tous les afficheurs LCDs, mais les caractères 16 à 31 & 128 à 255 peuvent variés suivants les différents fabricants. Donc plusieurs afficheurs LCDs peuvent affichés des caractères différents de ceux de la table.

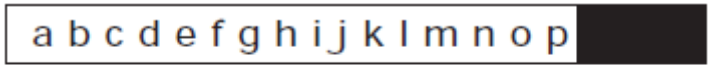
Les caractères 0 à 15 sont décrit comme des caractères « personnalisables » et ainsi peuvent être définis avant utilisation, ou ils contiennent des caractères de «forme aléatoire». Pour savoir en détail comment utiliser ces caractères voir les datasheets des fabricants de LCD.

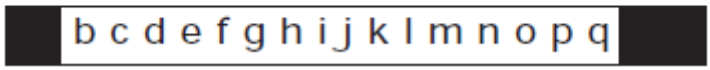
Le fonctionnement de l'affichage est assez complexe car l'affichage peut effectivement stocker plus de caractères que ceux qui peut être affiché à la fois. Un modèle simple peut le faire comprendre facilement. Imaginez un morceau de papier avec une rangée de lettre écrites dessus. Prenez un morceau de carte, qui a une fenêtre découpée, et placez la carte sur le papier, seules quelques lettres seront visibles. Les autres lettres sont toujours là, c'est juste qu 'elle ne peuvent pas être vues. C'est comme cela que l'afficheur LCD travaille – il stocke un lot de caractères, mais en montre seulement une partie à travers la fenêtre d'affichage

20 lettres stockées dans la mémoire de l'afficheur



Seulement, 16 caractères peuvent être vus en même temps





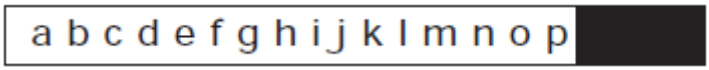
démarrer avec un morceau de papier, sur lequel une lettre est écrite. Placez la carte sur le papier, et la lettre sera visible parce qu'elle est montrée à travers la fenêtre de la carte. Enlevez la carte, écrivez une autre lettre, replacez la carte et elles seront visibles toutes les deux. En fait les 16 premières lettres seront visibles, mais la dix septième lettre ne sera pas visible, car la « fenêtre d'affichage » n'a qu'une capacité de 16 lettres.

Papier vierge

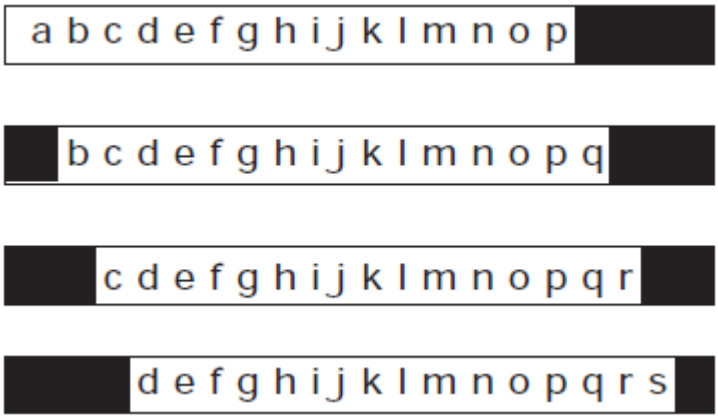
La première lettre peut être vue

la lette suivante peut être lue

La 17^{iéme} lettre ne peut pas être vue car elle est en dehors de la fenêtre



Pour pouvoir voir la 17^{iéme} lettre il est nécessaire de déplacer la fenêtre d'affichage d'une place sur la droite, maic ceci aussi signifie que la première lettre ne peut plus être vue. L'avantage de cette méthode de défilement devant une fenêtre est de faire apparaître de long message en le faisant dérouler sur l'écran LCD. Pour ce faire un message long est écrit dans la mémoire LCD, et ensuite la fenêtre d'affichage est déplacée la long du message. Ceci est équivalent à « pousser » le papier sous la fenêtre de la carte pour montrer le long message. La fenêtre LCD ne bouge pas « physiquement » - de sorte que tous ceux qui regardent l'écran verront les lettres se déplacer vers la droite.



Sur la plupart des afficheurs LCD il y a une mémoire de 40 caractères pour chaque ligne. Chaque espace dans la mémoire RAM peut être vue comme une boite qui est prête a emmagasiner un seul caractère. Chaque boite RAM a une adresse chiffrée pour la décrire. Les cases de la première ligne de la RAM ont l'adresse 128 à 191 et la seconde lignes de cases de 192 à 255.

Les afficheurs 16x2 ont une fenêtre qui a deux lignes. Ceci signifie que 16 lettres peuvent être vues sur chaque ligne. Si un caractère est pour être imprimé sur la seconde ligne, il est nécessaire de déplacer le curseur pour débuter sur la ligne 2. Le déplacement du curseur est très simple, envoyez simplement l' adresse RAM (de la case à déplacer) comme une instruction. Donc pour déplacer le curseur pour commencer la seconde ligne, envoyez simplement l'instruction « 192 » au module LCD. Pour déplacer le curseur à la cinquième position sur la seconde ligne, envoyez l'instruction « 197 » (=192+5).

Note au sujet des afficheurs 16x1

La plupart des LCDs 16x1 sont en fait actuellement des LCDs 8x2 mais avec la seconde ligne positionnée directement après la première (au lieu d'être en dessous). Cela rend l'affichage 16x1 confus à utiliser, car, après que 8 caractères ont été imprimés, le curseur semble disparaître au milieu de l'afficheur ! Si ce type d'afficheur est nécessaire, rappelez vous que le « neuvième » caractère est actuellement le premier de la seconde ligne.

Il y trois manières d'interfacer les LCDs avec les micro-contrôleurs

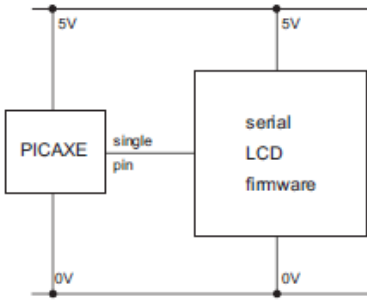
1) Puce logicielle LCD série
2) Module LCD série avec puce logicielle incorporée
3) Connexion Directe

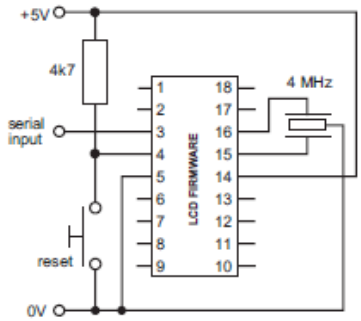
Connexion des LCDs utilisant une puce logicielle série (OPTION 1)

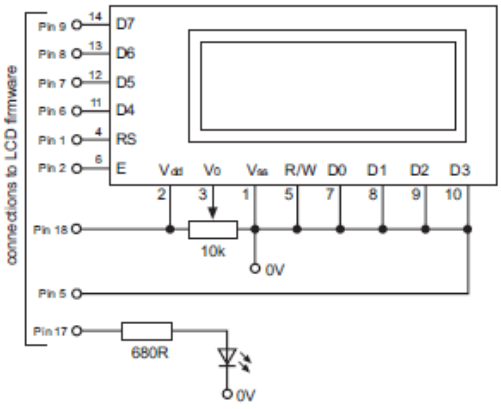
Le logiciel série LCD est utilisé pour permettre une commande série d'un LCD alphanumérique. Ceci permet aux micro-contrôleurs (et micro-contrôleur basés sur des systèmes tels que le PICAXE ou Basic Stamp) de visualiser les instruction d'utilisation ou de lire sur un écran texte sans avoir besoin d'avoir un ordinateur hôte. Ceci est spécialement utile lors de travaux avec des capteurs analogiques , car la lecture analogique peut aisément être affichée sur le module LCD. Toutes les commandes LCD sont transmises sériellement par une simple broche du micro-contrôleur. Un exemple d'instruction, utilisant la commande `serout` **:**

pour imprimer le texte « Hello » l'instruction est simplement :

```
serout 7, T2400, ' »Hello »)
```







Pour plus d'information, voir le datasheet du Serial LCD Firmware :

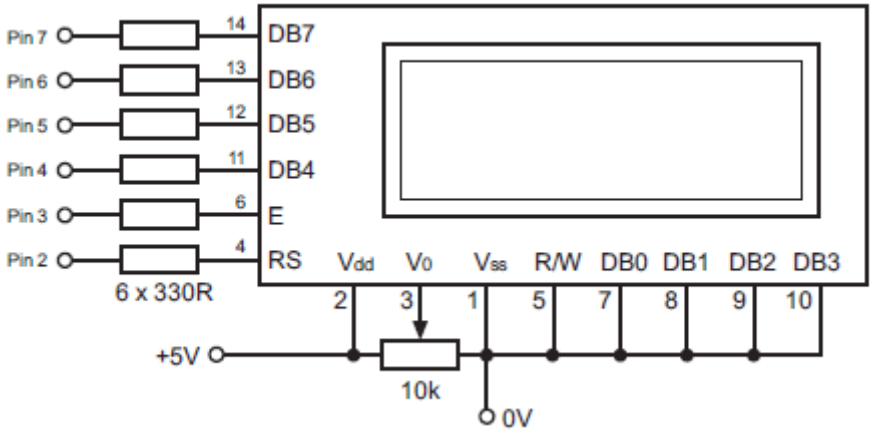
www.rev-ed.co.uk/docs/frm010.pdf

Utilisation d'un Module LCD série (OPTION 2)

Le module série LCD réf : AXE033 contient un module LCD monté sur un circuit imprimé équipé d'une puce logiciel LCD. Ceci permet à l'utilisateur de créer rapidement un circuit LCD qui utilise un simple fil de connexion comme avec l'option 1 ; voir le datasheet AXE033 Serial LCD/Clock Module pour plus de détails :

www.rev-ed.co.uk/docs/axe033.pdf

Connexion du LCD (OPTION 3)



Le LCD a 6 lignes qui peuvent être connectées directement aux broches du micro-contrôleur PICAXE . Toutefois il est conseillé d'ajouter une résistance de faible valeur (ex : 330 Ω) sur les lignes pour protéger contre les décharges électrostatiques. Le potentiomètre de 10k connecté sur la broche 3 est utilisé pour ajusté le contraste de l'affichage. Toutes les lignes non utilisées doivent être reliées à la masse.

Un Programme LCD Simple

Le programme suivant imprimera la phrase « Hello there ! » sur deux lignes d'un afficheur LCD. Il utilise trois sous-procédures appelées init, wrins et wrchr. Ces trois sous-procédures effectueront toutes les tâches logicielles «difficiles», et sont des sous-procédures «standards» qui non pas à être changées. En fait elles peuvent être utilisées sans comprendre comment elles travaillent, mais il est nécessaire de savoir ce qu'elles font.

init « initialise » le LCD ainsi il est prêt à recevoir les instructions.

wrins envoie une instruction stockée dans la variable b1 vers le module LCD.

wrchr envoie un caractère stocké dans la variable b1 pour être « imprimée » sur l'écran LCD.

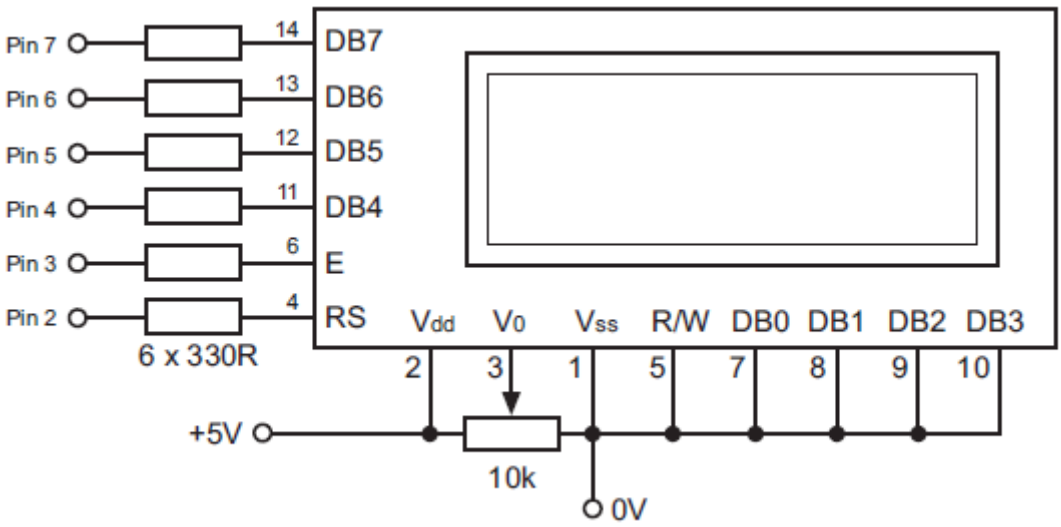
```
EEPROM 0, ' « Hello there ! »)        'stocke le texte dans la mémoire EEPROM

        gosub init              ' initialisation LCD

main :  let b1 = 1                    '  défini b1 pour instruction  « affichage clair »
        gosub wrins                   ' envoie l'instruction vers le LCD

        for b3 = 0 to 4               'démarre une boucle for…..next (« Hello » - position 0 à 4)
                read b3, b1           ' met la lettre depuis l'EEPROM dans la variable b1
                gosub wrchr           ' envoie le caractère au LCD
        next  b3                      ' boucle suivante

        let b1 = 192                  ' défini b1 pour commencer en position « seconde ligne »
        gosub wrins                   ' envoie l'instruction au LCD

        for b3 = 5 to 11              ' démarre une boucle for...next (« there ! » - position 5 à 11)
                read b3, b1           ' met la lettre depuis l'EEPROM dans la variable b1
                gosub wrchr           ' envoie le caractère au LCD
        next b3                       ' boucle suivante
```

Programme LCD plus Avancé

Le programme suivant déroule le message « Hello there everybody » ç travers l'écran. Comme le texte dépasse la longueur de 16 lettres, le message est d'abord stocké dans la mémoire LCD, et et ensuite la fenêtre d'affiche est déplacée successivement pour montrer tout le message.

```
EEPROM 0, (« Hello there everybody ! »)    ' charge le texte dans la mémoire EEPROM
        gosub init                         ' initialisation LCD

start :  let b1 = 1                        '  défini b1 pour instruction  « affichage clair »
        gosub wrins                        ' envoie l'instruction vers le LCD

        for b3 = 0 to 22                   'démarre une boucle for…..next
                read b3, b1                ' met la lettre depuis l'EEPROM dans la variable b1
                gosub wrchr                ' envoie le caractère au LCD
        next  b3                           ' boucle suivante

        let b1 = 12                        ' défini b1 pour l'instruction « curseur caché »
        gosub wrins                        ' envoie l'instruction au LCD

main: let b1 = 24                          ‘ défini b1 avec l'instruction « défilement affichage à gauche »
        gosub wrins                        ‘ Envoie l'instruction au  LCD
        pause 250                          ‘ pause pour 0.25s
        goto main                          ‘ boucle
```

Sous-Procédures Standard LCD (Connexion Directe)



Avant que que les sous-procédures soient étudiées, il est important de comprendre comment fonctionne le module LCD. Il a deux modes de fonctionnement, qui sont appelés mode « caractère » et mode « instruction ». La broche RS (broche 2) comme le mode **H**aut, le LCD est en mode caractère, **B**as le LCD est en mode instruction.

Le caractère ou instruction est envoyé comme un nombre binaire de 4 bits sur les lignes de données (broches 7-4). Chaque fois que la broche Validation (broche 3) est activée, le LCD lit les lignes de donnée et imprime le caractère (ou transmet l'instruction) qui est donné par le le nombre sur les lignes de donnée.

Ceci n'est pas tout à fait la réalité, car chaque caractère ou instruction est actuellement transmit sous forme d'un mot de 8 bits, qui contient une table de tous les codes de caractère ou d'instruction.Comme il n'y a que 4 lignes de donnée, les nombres de 8 bits sont partagés en deux « moitiés » qui sont envoyées l'une après l'autre. Les deux moitiés sont appelées « quartet haut » et « quartet bas ». Ceci signifie que deux quartet sont transmis sur les lignes de donnée pour chaque caractère.

1011		0101	= 10110101
quartet haut	+	quartet bas	= octet

Les trois sous-procédures standards décrites avant de réaliser tous les logiciels compliqués disent quand utiliser l'afficheur LCD. Chaque sous-procédure est appelée depuis le programme principal pour effectuer une certaine tâche. Ces tâches sont :

init initialise l'afficheur et mets le module en mode deux lignes

wrchr « imprime un « caractère » sur l'écran LCD

wrins écrit une commande dans le module LCD

(actuellement juste la sous-procédure wrchr en addition avec une ligne qui initialise la ligne RS en mode « instruction »au démarrage de la sous-procédure.

```
init:   let pins = 0            ‘ Efface toutes les lignes de sortie
        let b2 = 0              ‘ Réinitialise la variable b2
        let dirs = 252          ‘ Réinitialise les broches 2-7 comme lignes de sortie (Stamp
                                seulement).
        pause 200               ‘ Attend 200 ms pour réinitialiser le LCD.
        let pins = 48           ‘ Définir en mode  8-bit .
        pulsout 3,1             ‘ Envoie une donnée une impulsion sur « enable »
        pause 10                ' Attend 10 ms
        pulsout 3,1             ‘ Envoie encore la donnée
        pulsout 3,1             ‘ Envoie encore la donnée
        let pins = 32           ‘ Définir en mode  4-bit.
        pulsout 3,1             ‘ Envoie la donnée.
        pulsout 3,1             ‘ Envoier encore la donnée.
        let pins = 128          ‘ Définir en mode deux lignes
        pulsout 3,1             ‘ Envoie la donnée.
        let b1 = 14             ‘ Écran on, instruction curseur on
        gosub wrins             ‘ Écrit l'instruction dans le LCD
        return

wrchr:  let pins = b1 & 240     ‘ Masque le quartet haut de b1 dans b2.
        high 2                  ‘ Être sur que RS est haut
        pulsout 3,1             ‘ Impulsion sur la broche « enable » pour envoyer la donnée.
        let b2 = b1 * 16        ‘ Met le quartet bas de b1 dans b2
        let pins = b2 & 240     ‘ Masquer le quartet haut de b1 dans b2
        high 2                  ‘ S'assure que RS est haut
        pulsout 3,1             ‘ Impulsion sur la broche « enable » pour envoyer la donnée.
        return

wrins:  let pins = b1 & 240     ‘ Masque le quartet haut de b1 dans b2.
        pulsout 3,1             ‘ Impulsion sur la broche « enable » pour envoyer la donnée .
        let b2 = b1 * 16        ‘ Met le quartet bas de b1 dans b2 .
        let pins = b2 & 240     ‘ Masque le quartet haut de b2
        pulsout 3,1             ‘ Impulsion sur la broche « enable » pour envoyer la donnée.
        high 2                  ‘ Retour au mode caractère
        return
```

Notez que `init` *utilise des commandes* let dirs = *qui affecteront les 8 broches et pas seulement les 6 utilisées par l'afficheur LCD . Les commandes* let pins = *utilisées par* `wrins/wrchr` *ne modifient pas les broches 0 et 1 inutilisées . Ne pas utiliser les variables b1 ou b2 (ou w0 ou w1) pour d'autre fonction dans le programme.*

Utilisation du jeu d'instruction LCD

Les codes pour les instructions LCD sont donnés ci-dessous. Chaque code peut être envoyé au module LCD par l'utilisation de la sous-procédure `wrins`. Ces instructions peuvent être utilisées pour rendre le message LCD plus intéressants – par exemple, en faisant clignoter l'écran ou en créant des message se déplaçant à travers l'écran.

Code	Instruction
1	Efface l'écran et se déplace pour commencer à la première ligne
2	Déplace le curseur et la fenêtre d'affichage pour commencer à la première ligne
4	Définit le mode « impression de droite à gauche »
5	Définit le mode « impression déroulante vers la gauche »
6	Définit le mode « impression gauche à droite »
7	Définit le mode « impression déroulante vers la droite »
10	Met l'écran LCD sur Off
12	Cache le curseur
13	Fait flasher le curseur
14	Met l'écran LCD et (le curseur) sur On
16	Déplace le curseur d'une position à gauche
20	Déplace le curseur d'une position à droite
24	Faire défiler la «fenêtre» d'affichage à gauche d'une position
28	Faire défiler la «fenêtre» d'affichage à droite d'une position
128	Déplacer le curseur pour commencer sur la première ligne
192	Déplacer le curseur pour commencer sur la première ligne

Exemples :

Effacer l'afficheur

```
clear :     letb1 = 1                ' défini b1 pour effacer l'écran
            call wrins               ' l'envoie au LCD
```

Déplace le curseur à la seconde ligne

```
clear :         let b1 = 192         ' défini b1 pour commencer en seconde ligne
            call wrins               ' l'envoie au LCD
```

Flasher un message 10 fois

```
flash:      for b3 = 1 to 10         ‘ Commence une boucle for….next utilisant
                                     ‘ la variable b3.  N'utilisez pas b1!!
            let b1 = 10              ‘ Met dans b1 l'instruction « Met l'écran LCD
                                     ' sur Off »

            gosub wrins              ‘ Envoie l'instruction au LCD
            pause 200                ‘ Pause pour 0.2 seconde
            let b1 = 14              ‘ met dans b1 l'instruction « Met l'écran LCD
                                     ' sur On»
            gosub wrins              ‘ Envoie l'instruction au LCD
            pause 200 ‘ Pause for 0.2 second
        next b3                      ‘ Fin de la boucle  for...next
```

Déroulement d'un message (longueur 30 caractères)

```
scroll:     for b3 = 1 to 30         ‘ Commence une boucle for….next utilisant
                                     ‘ la variable b3.  N'utilisez pas b1!!
            let b1 = 28              ‘ Met dans b1 l'instruction  « Faire défiler
                                     ‘ la fenêtre d'affichage à droite d'une position »
            gosub wrins              ‘ Envoie l'instruction au LCD
            pause 200                ‘ Pause pour 0.2 seconde
        next b3                      ‘ Fin de la boucle  for...next
        let b1 = 1                   ‘ Met dans b1 l'instruction « Efface l'écran et se
                                     ' déplace pour commencer à la première ligne »
        gosub wrins                  ‘ Envoie l'instruction au LCD
        pause 200                    ‘ Pause pour 0.2 seconde
        goto scroll                  ‘ Boucle
```

Interfaçage Avancé 2 – Interfaçage série vers un Ordinateur

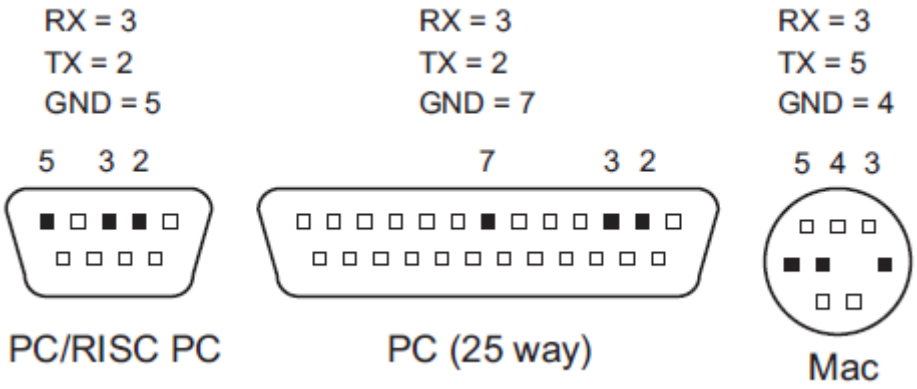
La plupart des ordinateurs peuvent dialoguer avec d'autres équipements par une communication série. La communication série utilise un « protocole » (ou code) où les caractères sont convertis en nombres et ensuite sont transmis par des câbles ? Une souris « communique » normalement sériellement avec un ordinateur et un modem travaille travaille en convertissant ces nombres en sons pour voyager sur les lignes téléphoniques.

Comme tous les ordinateurs utilisent le même code ASCII pour la transmission et la réception des caractères il est relativement facile de programmer le micro-contrôleur PICAXE pour discuter avec n'importe quel type d'ordinateur.Tout ce qui est nécessaire st une suite de câble et quelques circuits électronique très simples.

Connexion à l'ordinateur

Le système que nous voulons utiliser demande juste trois fils entre l'ordinateur et le micro-contrôleur. Le fil de masse fournit une référence commune ? Le fil RX envoie le signal de l'ordinateur au PICAXE et le fil TX envoie les signaux du micro-contrôleur PICAXE à l'ordinateur.

Le meilleur moyen de faire un câble série est d'acheter un câble rallonge série et de le couper en deux. Ceci vous donnera deux câbles adaptés avec un connecteur à chaque extrémité. Le diagramme ci-dessous vous montre les différentes connexions câblées requises.



Logiciel de Communication Ordinateur

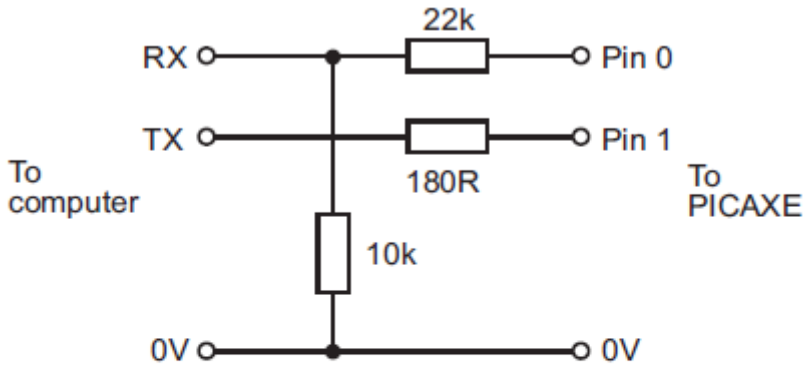
Pour utiliser ce système, un logiciel de communication est requis pour le PC. Les exemples ci-dessous utilise l'option Terminal au sein du logiciel PE6, mais n'importe quel logiciel de communication peut-être utilisé.

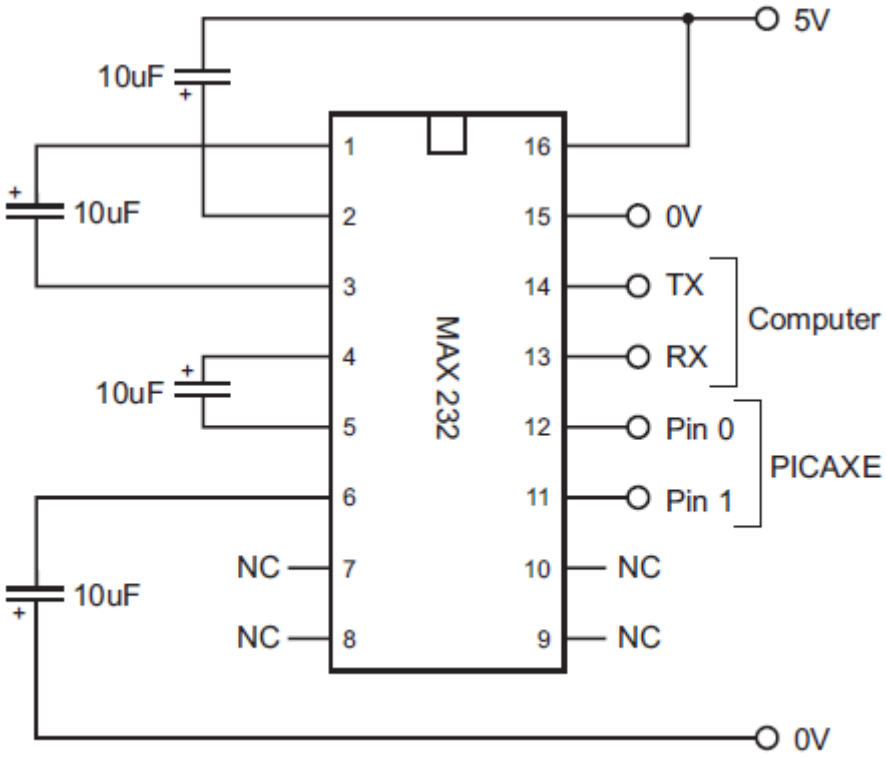
Il y a différents protocole qui peuvent être utilisés pour une communication série, et il est important que l'ordinateur et le micro-contrôleur aient les mêmes réglages. Le protocole 2400, N, 8,1 est utilisé ici : vitesse 2400 bauds, pas de parité, 8 bits de donnée et un bit de stop. Cette vitesse est lente pour un standard moderne, mais elle est suffisante pour la majorité des projets. Tous les « handshaking » (logiciel ou matériel) doivent être désactivé.

Circuit d'interfaçage pour Microcontrôleur PICAXE

Le système décrit ici demande juste 3 fils entre l'ordinateur et le micro-contrôleur PICAXE. Strictement parlant, la tension d'une RS232 devrait être de ± 15V mais le 5V standard du régulateur du circuit PICAXE sera utilisé ici. Ce n'est pas le standard de l'industrie mais il fonctionne très bien avec la plupart des ordinateurs. C'est le circuit qui doit être utilisé pour une communication série.

Pour fournir une vraie tension RS232 un autre circuit intégré est requit. LE CI le plus utilisé est le MAX232, qui a un ampli de tension pour créer la tension nécessaire. Si ceci est urilisé, il est nécessaire de changer le paramètre N2400 (négatif) dans tous les logiciels de communications série par T2400 (positif vrai). N'UTILISEZ QU'UN DE CES DEUX CIRCUITS





Remarque. Noter la polarité – les condensateurs connectés aux broches 2 et 6 sont connectés « à l'envers »

Transmission de Caractères à l'Écran de l'Ordinateur

Le programme suivant transmettra le mot « Hello » à l'écran de l'ordinateur encore et encore.Si le câble est connecté et que le logiciel de communication est opérationnel, le mot apparaîtra toutes les secondes.

```
main:   serout 1,N2400,(“Hello”)        ‘ Envoie le mot « Hello »
        serout 1,N2400,(10,13)          ‘ envoie l'instruction « nouvelle ligne »
        pause 1000                      ‘ attend une seconde
        goto main                       ‘ boucle vers le départ
```

Notez que "texte" doit être enfermé dans les « xx ». Ceci disent au micro-contrôleur de convertir le texte en une chaîne de codes ASCII. Les codes ASCII individuels peuvent être envoyés en envoyant juste leurs nombres. Toutefois les deux commandes ci-dessous réalisent la même tâche :

```
serout 1,N2400, (« Hello »
serout 1,N2400,( 72,101,108,108,111)
```

Recevoir les entrées clavier depuis l'ordinateur

Il peut être utile de pouvoir utiliser un clavier pour que les personnes « répondent » aux questions. Ceci est réalise en utilisant la commande `serin` comme montré ci-dessous :

```
main:   serout 1,N2400,(10,13)          ‘ Commence une nouvelle ligne
        serout 1,N2400,(“Press a key- “) ‘ Envoie un message
        serin 0,N2400,b1                ‘ Recoit un caractère dans la variable b1
        serout 1,N2400,(b1)             ‘ Transmet le caractère à l'écran
        if b1=”a” then hot              ‘ Est-ce le caractère « a » ? oui allez à hot
        goto main                       ‘ Non, boucle de retourt
hot:    serout 1,N2400, (10,13,”A is the Hot Key!”)
                                        ‘ Envoie message
        goto main                       ‘ Boucle de retour
```

Si ce programme est en service quand une touche est pressée au clavier, le caractère apparaît sur l'écran. C'est le micro-contrôleur (pas l'ordinateur) qui travaille. L 'action sur le clavier a été reçue depuis celui-ci et ensuite retransmise à l'écran !

Caractères ou nombres

Considerez cette commande : `serout  1, N2400, (65)`
Ceci enverra le caractère ASCII « A » à l'écran.

Maintenant Considerez la commande : `serout  1, N2400,(b1)`
Ceci enverra le caractère stocké dans la variable b1 à l'écran, et ainsi si b1 = 65, le caractère « A » sera envoyé à l'écran.

Toutefois, les variables sont souvent utilisées pour stocker les résultats de sommes mathématiques, et ainsi il peut être nécessaire d'envoyer le nombre « 65 » à l'écran plutôt que la lettre « A » . Pour ce faire, le micro-contrôleur doit être averti qu'un nombre va être envoyé plutôt qu'un caractère. Ceci est r&éalisé en ajoutant un dièse (#) :
`serout  1, N2400, (#b1)`

Ceci enverra le nombre « 65 »(actuellement les deux caractères « 6 » et « 5 ») à l'écran plutôt que le caractère « A ».

Ceci est un sommaire des commandes série utilisées. Rappelez-vous que le numéro de la broche peut avoir été changé, et également la section N2400 à T2400 si le circuit d'interface MAX232 est utilisé.

seront 1,N2400,(“Hello”)- Envoie un message à l'écran.
serout 1,N2400,(10) - Envoie une instruction directe ASCII à l'écran.
serout 1,N2400,(b1) - Envoie un un caractère ASCII stockée dans une variable à l'écran
serout 1,N2400,(#b1) - Envoie un un nombre stocké dans une variable à l'écran
serin 0,N2400,b1 - Reçoit un caractère ASCII d'une touche activée depuis le clavier et le stocke comme une valeur ASCII dans la variable (b1)
serin 0,N2400,#b1 – Reçoit un chiffre réel depuis le pavé numérique sur le clavier et le stocke dans une variable (b1)